TUGAS AKHIR – KI141502

STEGANOGRAFI TEKS PADA AKSARA SUNDA DENGAN PENDEKATAN FEATURE CODING

Mohammad Rijal
5113100193

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

STEGANOGRAFI TEKS PADA AKSARA SUNDA DENGAN PENDEKATAN FEATURE CODING

MOHAMMAD RIJAL
5113100193

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT - KI141502

TEXT STEGANOGRAPHY ON SUNDANESE SCRIPT USING FEATURE CODING APPROACH

MOHAMMAD RIJAL
5113100193

Supervisor I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Supervisor II
Tohari Ahmad, S.Kom., MIT., Ph.D.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
Sepuluh Nopember Institute of Technology
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

STEGANOGRAFI TEKS PADA AKSARA SUNDA DENGAN PENDEKATAN FEATURE CODING

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
MOHAMMAD RIJAL
NRP: 5113 100 193

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Henning Titi Ciptaningtyas, S.Kom., M.Kom.
(NIP. 198407082010122004) ..... (Pembimbing 1)

Tohari Ahmad, S.Kom., MIT., Ph.D.
(NIP. 197505252003121002) ..... (Pembimbing 2)

SURABAYA
Juli, 2017

[*Halaman ini sengaja dikosongkan*]

STEGANOGRAFI TEKS PADA AKSARA SUNDA DENGAN PENDEKATAN FEATURE CODING

Nama Mahasiswa : Mohammad Rijal
NRP : 5113 100 077
Jurusan : Teknik Informatika, FTIf ITS
Dosen Pembimbing 1 : Henning Titi Ciptaningtyas, S.Kom., M.Kom.
Dosen Pembimbing 2 : Tohari Ahmad, S.Kom., MIT., Ph.D.

ABSTRAK

Kemajuan teknologi membuat arus informasi semakin meningkat. Kebutuhan akan transmisi data yang aman menjadi sangat penting. Dalam dunia sekuritas, dikenal istilah steganografi. Steganografi merupakan suatu cara untuk menyembunyikan informasi tanpa diketahui keberadaan informasi rahasia tersebut. Salah satu jenis steganografi adalah steganografi teks yang menggunakan teks sebagai media penyampaian. Steganografi teks telah banyak digunakan pada pada aksara daerah dan tradisional di dunia, seperti Devanagari, Arab, Persia dan Telugu. Aksara Sunda yang memiliki 13 rarangkén atau vokalisasi dapat digunakan sebagai media dalam steganografi teks.

Metode yang ditawarkan pada tugas akhir ini menggunakan pendekatan *Feature Coding* pada aksara Sunda. Feature Coding mengubah bagian dari teks yang menjadi media untuk menyimpan bit informasi. Pada aksara Sunda, rarangkén atau vokalisasi digeser untuk menyimpan bit *secret message* yang menjadi masukan. Pilihan pergeseran pada tugas akhir ini terbagi menjadi *shift all* yang menggeser semua rarangkén dan *shift group* yang menggeser kelompok posisi rarangkén dalam satu aksara ngalagena.

Hasil uji coba metode menunjukkan bahwa steganografi teks pada aksara Sunda berhasil dilakukan dengan menggeser rarangkén melalui penggantian blok Unicode. *Capacity ratio* yang dihasilkan dipengaruhi oleh jumlah rarangkén yang ada. Pilihan

shift all menghasilkan rata-rata *capacity ratio* 57,01 *bit/kiloByte*. Pilihan *shift group* menghasilkan rata-rata *capacity ratio* yang lebih besar yaitu 60,74 *bit/kiloByte*. Kenaikan sebesar 6,54% disebabkan oleh jumlah rarangkén pada pilihan *shift group* yang lebih besar dari pilihan *shift all*.

Kata kunci: steganografi, aksara Sunda, feature coding, secret message, capacity ratio, rarangkén.

TEXT STEGANOGRAPHY ON SUNDANESE SCRIPT USING FEATURE CODING APPROACH

Student Name	: Mohammad Rijal
Registration Number	: 5113 100 193
Department	: Informatics Engineering, FTIf ITS
First Supervisor	: Henning Titi Ciptaningtyas, S.Kom. M.Kom
Second Supervisor	: Tohari Ahmad, S.Kom., MIT., Ph.D.

ABSTRACT

The rapid growth of technology makes the flow of information increasing. The need for secure data transmission becomes very important. In the digital securities field, the term steganography is widely known. Steganography is a way to hide information without being notified of the existence of such confidential information. One of the type of steganography is text steganography that uses text as a medium of transmission. Text steganography have been widely used in regional and traditional script, such as Devanagari, Arabic, Persian, and Telugu. Sundanese script which has 13 rarangkén or diacritics can be used as an alternative for steganography media.

The method proposed in this final project uses feature coding approach on Sundanese script. Feature Coding modify a part of letter for storing bits of information. In the Sundanese script, rarangkén or diacritics are shifted to store the secret message from input. The choice of shift is divided into shift all which shifts all rarangkén and shift group which shifts only rarangkén on the same group in a letter (ngalagena script).

The results of the method show that the text steganography on Sundanese script is done by shifting the rarangkén through replacement of Unicode blocks. Resulting capacity ratio is affected by the total number of rarangkén. The shift all option result an

average capacity ratio of 57.01 bit/kilobyte. The shift group option result a larger average capacity ratio of 60.74 bit/kilobyte. Increasing number of capacity by 6.54% is caused by the bigger number of rarangkén that shift group option has, compared to shift all option.

Key words: steganography, Sundanese script, feature coding, secret message, capacity ratio, rarangkén.

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"Steganografi Teks pada Aksara Sunda dengna Pendekatan Feature Coding".**

Buku tugas akhir ini disusun dengan harapan dapat memberikan manfaat dalam penelitian steganografi teks pada aksara daerah lebih lanjut. Selain itu, penulis berharap dapat memberikan kontribusi positif bagi kampus Teknik Informatika ITS dan masyarakat luas. .

Dalam perancangan, pengerjaan, dan penyusunan tugas akhir ini, penulis banyak mendapatkan bantuan dari berbagai pihak. Penulis ingin mengucapkan terima kasih kepada:

1. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom., dan Bapak Tohari Ahmad, S.Kom., MIT., Ph.D. selaku dosen pembimbing penulis yang telah memberi ide, nasihat, saran, dan bimbingan kepada penulis hingga penulis dapat menyelesaikan buku tugas akhir ini.
2. Orang tua penulis, Ibu Hj. Nunuy Nurhayati dan alm H. Dudung Abdul Halim yang tak henti-hentinya mendukung baik secara finansial maupun moral dari sejak kecil hingga penulis dapat menempuh jenjang perkuliahan.
3. Seluruh saudara kandung: A Cecep, A Ade, A Opik, Imen, dan Agil dan Keluarga Besar di Pondok Pesantren Cipasung yang telah menjadi saudara yang selalu pengertian dan memberikan nasihat kepada penulis.
4. M. Thoriq Aziz dan Mulya Maulana Irham yang selalu memberikan semangat dan menjadi tempat berbagi keluh kesah bagi penulis.
5. Kang Ilham Nurwansah yang telah memberikan penjelasan seputar aksara Sunda Unicode.

6. Teman-teman seperjuangan dan para admin Lab NCC yang telah banyak membantu memfasilitasi dalam pengerjaan serta menemani penulis dalam pengerjaan
7. Mahasiswa angkatan 2013 yang telah menghabiskan waktu bersama penulis selama empat tahun dalam suka maupun duka.
8. Pihak-pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari masih ada kekurangan dalam penyusunan tugas akhir ini. Penulis mohon maaf atas kesalahan, kelalaian maupun kekurangan dalam penyusunan tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan ke depan.

Surabaya, Juli 2017

Penulis

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Kemajuan teknologi membuat arus informasi semakin meningkat. Kebutuhan akan transmisi data yang aman menjadi sangat penting. Dalam dunia sekuritas, dikenal istilah steganografi dan kriptografi. Steganografi merupakan suatu cara untuk menyembunyikan informasi tanpa diketahui keberadaan informasi rahasia tersebut. Media yang disisipi informasi rahasia dalam steganografi tak akan terlihat menampilkan perbedaan [1]. Berbeda dengan steganografi, kriptografi menampilkan pesan yang acak dalam penyembunyian informasi sehingga lebih mudah untuk diketahui keberadaannya [2].

Steganografi secara umum terbagi menjadi steganografi gambar, audio, video, dan teks. Dari jenis tersebut, Steganografi Teks merupakan salah satu yang paling sulit disebabkan tingkat redundansi informasi yang rendah jika dibandingkan gambar atau audio [3]. Pada steganografi teks, penyembunyian informasi dapat digunakan pada semua bahasa. Shirali-Shahreza [4] memanfaatkan huruf yang mempunyai titik pada bahasa Arab untuk menyembunyikan bit informasi. Posisi titik pada huruf akan digeser sesuai dengan bit yang disembunyikan. Huruf bertitik dalam bahasa Arab yang berjumlah 13 membuat *hidden capacity ratio* yang dihasilkan tinggi. Penggunaan ekstensi dalam bahasa Arab dimanfaatkan oleh Gutub dan Fattani [5] untuk menyembunyikan bit informasi. Setiap huruf setelah huruf bertitik, penulisannya akan diberi ekstensi atau dipanjangkan. Metode ini mengatasi masalah *robustness* dari Shirali-Shahreza, tetapi terbatas pada kapasitas informasi karena tidak semua huruf

Arab dapat mempunyai ekstensi. Aabed dkk [6] menggunakan Diakritik atau Vokalisasi yang bisa digunakan pada setiap huruf Arab. Ketiga metode tersebut merupakan metode dasar yang bervariasi pada perkembangannya. Variasi tersebut dapat dilihat dari *Kashida Variation Algorithm* dan *Multipoint Letter Algorithm* oleh Odeh [7] [8] yang memungkinkan penyimpanan 2 bit dalam satu huruf.

Steganografi teks yang menggunakan huruf non-Latin tidak terbatas pada huruf Arab. Alla dan Prasad [9] memanfaatkan diakritik dan *compound words* pada huruf Devanagari, huruf yang digunakan pada bahasa India. Sementara itu Changder dkk [10] menggunakan pegeseran Matra, vokalisasi yang digunakan pada konsonan dalam huruf Devanagari, untuk menyimpan bit informasi. Pada metode yang ditawarkan Alla dan Prasad [11], posisi matra yang berada di atas atau di bawah digunakan untuk menandai bit "0" atau "1". Selain itu, Alla dan Prasad [12] mengembangkan metode steganografi dalam bahasa lokal di India yaitu Telugu dengan menggunakan kesesuaian *ottulu* dan tanda baca.

Aksara Sunda sebagai aksara Nusantara bersifat silabik, yaitu satu huruf mewakili nilai satu suku kata. Bunyi vokal dalam aksara Sunda dapat diganti dengan menggunakan *rarangkén*, yaitu vokalisasi atau diakritik yang ditulis di atas, di bawah, atau pun sejajar dengan huruf. [13] Dengan jumlah *rarangkén* yang mencapai 13 buah, aksara Sunda dapat digunakan sebagai alternatif media untuk steganografi teks.

Pada tugas akhir ini, penulis menggunakan Aksara Sunda sebagai media untuk steganografi teks. Metode yang ditawarkan menggunakan pendekatan *Feature Coding*, di mana terdapat perubahan posisi dari *rarangkén* setiap kali menyimpan bit informasi. Dengan jumlah *rarangkén* yang tinggi dan hampir dapat dipakai pada setiap huruf, hal ini diharapkan dapat meningkatkan *capacity ratio* dari steganografi teks. Untuk menjaga *robustness*, setelah selesai disisipkan informasi, maka teks akan dikonversi ke dalam bentuk PDF.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana melakukan proses penyembunyian informasi (*embedding message*) dalam steganografi teks pada aksara Sunda?
2. Bagaimana melakukan proses ekstraksi informasi (*extracting message*) dalam steganografi teks pada aksara Sunda?
3. Bagaimana menghitung performa hasil steganografi teks pada aksara Sunda?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aksara Sunda yang digunakan adalah aksara Sunda baku.
2. *Font* yang digunakan untuk modifikasi merupakan *font* Sundanese Unicode 2013.
3. Masukan *cover text* berupa aksara Sunda.
4. Masukan *cover text* bersifat manual dan tidak melalui proses *generate* otomatis.
5. Metode diimplementasikan menggunakan bahasa pemrograman Java.

1.4 Tujuan Tugas Akhir

Tujuan tugas akhir ini adalah melakukan proses steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding*.

1.5 Manfaat Tugas Akhir

Manfaat dari tugas akhir ini adalah menghasilkan sistem untuk melakukan transmisi informasi yang aman melalui proses steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding*.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

   Pada studi literatur, dilakukan pengumpulan data dan studi terhadap sejumlah referensi yang diperlukan dalam pengerjaan tugas akhir. Referensi tersebut didapatkan dari beberapa artikel yang dipublikasikan oleh jurnal. Selain dari artikel, studi literatur juga dilakukan melalui pencarian referensi dari internet yang membahas mengenai informasi yang dibutuhkan, seperti steganografi, metode feature coding, dan aksara Sunda.

2. Analisis dan Desain Perangkat Lunak

   Pada tahap ini disusun perancangan dari sistem steganografi teks pada aksara Sunda yang dibangun. Perancangan ini berisi tentang alur kerja sistem dan data yang digunakan. Selain itu, terdapat desain antarmuka dari sistem yang akan dibangun.

3. Implementasi Perangkat Lunak

   Sistem steganografi teks pada aksara Sunda akan dibuat dengan bahasa pemrograman Java menggunakan perangkat pengembangan NetBeans IDE pada platform *desktop*. *Library* yang digunakan pada sistem yaitu Apache POI dan Apache PDFBox. Untuk modifikasi *font* Sundanese Unicode 2013, digunakan editor FontForge.

4. Uji Coba dan Evaluasi

   Dalam tahap ini, dilakukan pengujian sistem menggunakan parameter yang berbeda. Sistem akan diuji dengan data, *shifting option*, dan bit penanda panjang *secret message* yang berbeda.

1.7 Sistematika Laporan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini berisi tentang analisis dan perancangan desain sistem steganografi teks pada aksara Sunda.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini membahas tahap-tahap uji coba. Kemudian hasil uji coba dievaluasi untuk kinerja dari aplikasi yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

[*Halaman ini sengaja dikosongkan*]

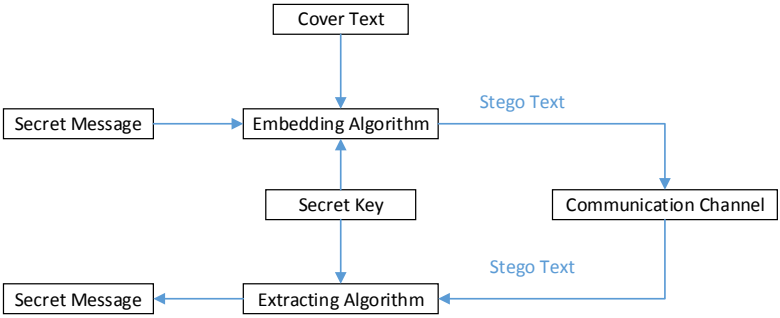
BAB II
TINJAUAN PUSTAKA

Pada bab ini berisi penjelasan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir dengan tujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan.

2.1 Steganografi

Steganografi berasal dari bahasa Yunani, *Steganos* yang berarti "tertutup", dan *Graphein* yang berarti "penulisan". Secara harafiah, steganografi berarti sistem penulisan yang tertutup (tersembunyi). [14] Dalam istilah teknologi informasi modern, steganografi diartikan sebagai sebuah metode untuk menyembunyikan informasi tanpa diketahui keberadaannya. Pesan rahasia akan dimasukan ke dalam sebuah media dengan cara keberadaan informasi rahasia tersebut disembunyikan. Tujuan steganografi adalah membuat sarana komunikasi yang tak diketahui keberadaannya dan menghindari kecurigaan dalam pengiriman pesan rahasia.

Pada steganografi umumnya terdiri dari tiga unsur utama yaitu: pesan rahasia (*secret message*), *cover object*, *stego object*. Pesan rahasia merupakan pesan yang akan dikirimkan dan disisipkan ke *cover object*. Pesan yang dikirimkan umumnya tidak terlalu panjang dan menyesuaikan dengan bit yang dapat ditampung oleh *cover object. Cover object* merupakan media yang digunakan untuk menyembunyikan pesan rahasia yang akan dikirim. Media yang digunakan terlihat normal dan dapat berupa gambar, audio, video, atau teks. Metode penyembunyian informasi ke *cover object* bervariasi tergantung media yang digunakan. Setelah disisipikan pesan rahasia, *cover object* berubah menjadi *stego object*. Pada dasarnya *stego object* akan terlihat sama dengan *cover object*. Selain itu, terkadang terdapat *secret key* yang menjadi kunci untuk menyembunyikan dan mengembalikan informasi [15].



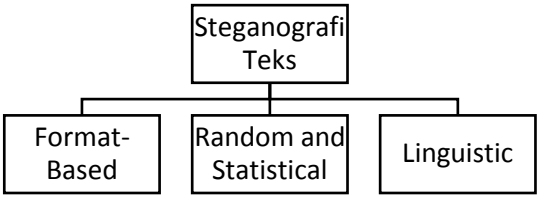
Gambar 2.1 Alur Penyembunyian Informasi pada Steganografi

Dalam prosesnya, steganografi terdiri dari proses penyisipan informasi, *embedding*, dan pengembalian informasi, *extracting*. Alur proses ditunjukkan pada Gambar 2.1, Pada proses *embedding*, pesan rahasia akan disisipkan pada *cover object* yang kemudian menjadi *stego object*. Setelah itu, *stego object* akan dikirimkan melalui media komunikasi kepada tujuan. Setelah diterima oleh tujuan, akan dilakukan proses *extracting* pada *stego object*. Proses *extracting* akan melakukan ekstraksi informasi yang terdapat pada *stego object* [16].

Pada tugas akhir ini steganografi dipilih sebagai cara penyembunyian informasi. Unsur *cover object* dan *stego object* yang digunakan adalah teks aksara Sunda. Sementara itu, *secret key* berupa huruf Sundanese Unicode 2013 hasil modifikasi dan pilihan *shifting option*.

2.2 Steganografi Teks

Steganografi teks adalah salah satu jenis steganografi yang menggunakan teks sebagai media penyembunyian informasi. Metode dalam steganografi teks dapat diklasifikan menjadi tiga metode utama yaitu *Format-Based Methods, Random and Statistical Generation Methods*, dan *Linguistic Methods* seperti ditunjukkan pada Gambar 2.2. [7].



Gambar 2.2 Metode dalam Steganografi Teks

Format-Based Methods dalam steganografi teks mengubah secara fisik format dari teks untuk menyembunyikan informasi. Tidak ada perubahan dalam teks secara isi kalimat dan makna. Contoh *Format-Based Methods* yang sederhana adalah metode *Open Spaces* yang menambahkan spasi dalam teks. Untuk setiap penambahan satu spasi akan diterjemahkan sebagai bit "0" dan dua spasi yang berurutan diterjemahkan sebagai bit "1". Selain metode tersebut, terdapat metode *Word-shifting*, *Line-shifting*, dan *Feature Coding* [7].

Dalam *Random and Statistical Generation Methods*, penyembunyian informasi dilakukan dengan menghasilkan *cover text* secara otomatis yang sesuai dengan properti statistik dari sebuah bahasa. Metode ini biasanya menggunakan grammar untuk menghasilkan *cover text* dalam bahasa tertentu. Contoh yang paling populer adalah penggunaan *Probabilistic Context Free Grammar (PCFG)* [7].

Metode terakhir adalah *Lingustic Methods* yang memanfaatkan unsur linguistik dari suatu bahasa. Dalam *Linguistic Methods*, terdapat *Synctatic Methods* dan *Semantic Methods*. *Syntatic Methods* menggunakan tanda pungtuasi (tanda baca) yang diletakan di tempat tertentu dalam teks untuk menyimpan bit informasi. Sementara itu penyembunyian informasi dalam *Semantic Methods* dilakukan dengan penggantian suatu kata dengan sinonimnya [7].

Pada tugas akhir ini steganografi teks dengan metode yang *Format-Based Methods* dipilih karena proses pergeseran teks aksara Sunda sebagai media, mengubah teks secara fisik.

2.3 Feature Coding

Feature Coding adalah metode steganografi teks di mana bit informasi disimpan dengan cara merubah satu atau beberapa bagian dari teks secara fisik. Contoh penggunaan *Feature Coding* adalah posisi titik pada huruf latin "i' dan "j' yang bisa digeser, panjang dari garis vertical dalam huruf latin "f" dan "t" yang dapat dirubah, atau dengan memanjangkan atau memendekkan tinggi dari huruf "b", "d" atau "h". [17]

Gambar 2.3 Feature Coding pada Huruf Nun [4]

Penggunaan *Feature Coding* pada aksara Tradisional terdapat pada steganografi teks pada huruf Arab dan Persia yang dirawarkan oleh Shirali-Shahreza. Shirali-Shahreza memanfaatkan huruf yang memiliki titik pada huruf Arab dan Persia untuk menyembunyikan informasi. Pada Gambar 2.3, titik pada huruf Nun digeser ke atas untuk menyembunyikan satu bit informasi. [4]

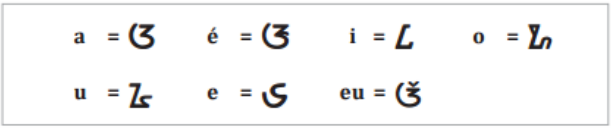
Pada tugas akhir ini, pendekatan *Feature Coding* digunakan pada rarangkén aksara Sunda. Posisi rarangkén akan digeser untuk menyimpan bit informasi rahasia. Rarangkén yang digeser menandakan bit "1", sedangkan rarangkén yang tidak digeser menandakan bit "0".

2.4 Aksara Sunda

Berdasarkan Peraturan Daerah Tingkat 1 Jawa Barat, aksara Sunda adalah sistem ortografi hasil kreasi masyarakat Jawa Barat yang meliputi aksara dan sistem pengaksaraan untuk menuliskan bahasa Sunda. Aksara Sunda berjumlah 32 buah yang terdiri atas 7 aksara swara 'vokal mandiri' (a, é, i, o, u, e, dan eu) dan 23 aksara ngalagena 'konsonan' (ka-ga-nga, ca-ja-nya, ta-da-na, pa-ba-ma, ya-ra-la, wasa-ha, fa-va-qa-xa-za) [13]. Pada tugas akhir ini, teks

aksara Sunda digunakan sebagai media untuk menyembunyikan informasi.

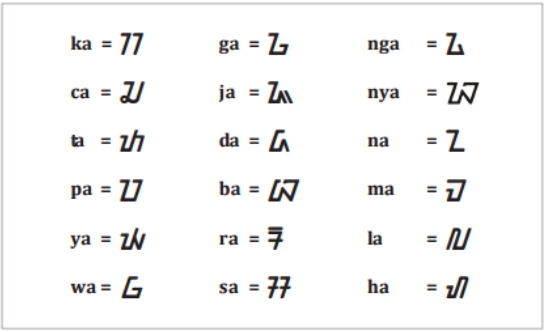
2.4.1 Aksara Swara



Gambar 2.4 Aksara Swara [13]

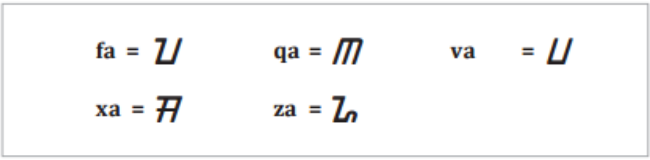
Aksara swara adalah tulisan yang melambangkan bunyi fonem vokal mandiri yang dapat berperan sebagai sebuah suku kata yang bisa menempati posisi awal, tengah maupun akhir sebuah kata. Aksara swara berjumlah 7 aksara yang ditunjukkan pada Gambar 2.4. [13]

2.4.2 Aksara Ngalagena



Gambar 2.5 Aksara Ngalagena dari Bunyi Bahasa Sunda [13]

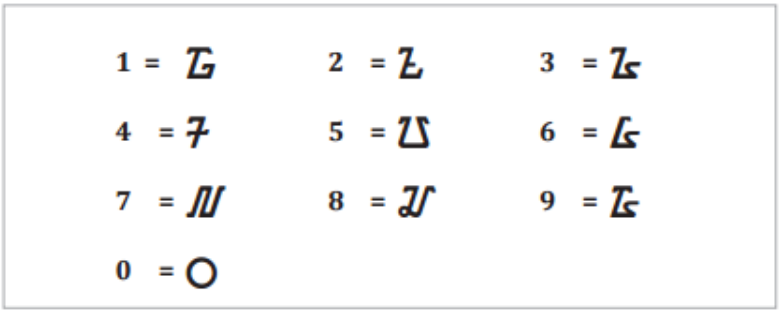
Aksara ngalagena adalah tulisan yang secara silabis dianggap dapat melambangkan bunyi fonem konsonan dan dapat berperan sebagai sebuah kata maupun suku kata yang bisa menempati posisi awal, tengah maupun akhir sebuah kata.



Gambar 2.6 Aksara Ngalagena dari Bunyi Serapan [13]

Pada awalnya aksara ngalagena dalam sistem tata tulis aksara Sunda Kuno berjumlah 18 buah seperti ditunjukkan pada Gambar 2.5. Namun, dalam upaya memenuhi fungsi aksara Sunda sebagai alat rekam bahasa Sunda yang senantiasa berkembang akibat terjadinya proses serapan unsur kosa kata asing, maka para pakar di bidang paleografi Sunda dan pihak birokrat di lingkungan Provinsi Jawa Barat beserta para tokoh masyarakat sepakat untuk mengaktifkan 5 lambang aksara yang ditunjukkan pada Gambar 2.6 ke dalam sistem tata tulis aksara Sunda Baku, sehingga jumlahnya menjadi 23 buah. [13]

2.4.3 Angka



Gambar 2.7 Angka dalam Aksara Sunda [13]

Sistem tata tulis aksara Sunda dilengkapi pula dengan lambang angka-angka seperti ditunjukkan pada Gambar 2.7. Penulisan lambang angka puluhan, ratusan, dan seterusnya ditulis berderet dari “kiri ke kanan”, seperti halnya dalam sistem angka Arab. Beberapa lambang angka Sunda bentuknya ada yang mirip

dengan lambang aksara sehingga untuk menuliskan (deretan) lambang angka harus diapit dengan garis vertikal yang lebih tinggi dari lambang angka. [13]

2.4.4 Pungtuasi

Pungtuasi atau tanda baca yang dipakai untuk melengkapi penggunaan aksara Sunda dalam penulisan suatu kalimat, alinea, maupun wacana dilakukan dengan mengadopsi semua tanda baca yang berlaku pada sistem tata tulis huruf Latin. Tanda baca yang dimaksud adalah koma (,), peun 'titik' (.), titik-koma (;), deubeul peun 'titik-dua' (:), panyeluk 'tanda seru' (!), pananya 'tanda tanya' (?), kekenteng 'tanda kutip' ("‥"), panyambung 'tanda hubung' (-), tanda kurung(()), dan sebagainya. [13]

2.5 Rarangkén

Dalam sistem tata tulis aksara Sunda dikenal adanya tanda vokalisasi, yaitu rarangkén atau penanda bunyi yang dapat berfungsi untuk mengubah, menambah maupun menghilangkan bunyi vokal pada aksara ngalagena. Lambang vokalisasi yang dimaksud berjumlah 13 macam yang dalam penempatannya terbagi ke dalam tiga kelompok. Kelompok pertama, sebanyak 5 buah yang ditempatkan di atas aksara dasar. Kelompok kedua, sebanyak 3 buah yang ditempatkan di bawah aksara dasar. Kelompok ketiga, sebanyak 5 buah yang ditempatkan sejajar dengan aksara dasar, yang dibagi lagi menjadi: 1 buah ditempatkan di sebelah kiri aksara dasar, 2 buah ditempatkan di sebelah kanan aksara dasar, dan sebanyak 2 buah ditempatkan di sebelah kanan dengan sedikit menjulur ke bagian bawah aksara dasar. Untuk aksara swara, rarangkén yang dapat digunakan hanya panglayar, panyecek, dan pangwisad. [13]

Pada tugas akhir ini rarangkén digunakan untuk menyembunyikan bit informasi. Posisi rarangkén akan digeser untuk menandakan bahwa bit informasi telah disispkan pada rarangkén tersebut.

2.5.1 Rarangkén Atas

-  = Panghulu, berfungsi merubah bunyi vokal aksara dasar /a/ menjadi /i/.
-  = Pamepet, berfungsi merubah bunyi vokal aksara dasar /a/ menjadi /e/.
-  = Paneuleung, berfungsi merubah bunyi vokal aksara dasar /a/ menjadi /eu/.
-  = Panglayar, berfungsi menambah konsonan /+r/ pada akhir aksara dasar.
-  = Panyecek, berfungsi menambah konsonan /+ng/ pada akhir aksara dasar. [13]

2.5.2 Rarangkén Bawah

-  = Panyuku, berfungsi merubah bunyi vokal aksara dasar /a/ menjadi /u/
-  = Panyakra, berfungsi menambah bunyi aksara /+ra/ pada aksara dasar yang didekatinya, dan bisa disesuaikan dengan tanda vokalisasi pada aksara dasarnya.
-  = Panyiku, berfungsi menambah bunyi aksara /+la/ pada aksara dasar yang didekatinya, dan bisa disesuaikan dengan tanda vokalisasi pada aksara dasarnya. [13]

2.5.3 Rarangkén Samping (Sejajar)

-  = Panéléng, berfungsi mengubah bunyi vokal aksara dasar /a/ yang didahuluinya menjadi /é/.

-  = Panolong, berfungsi mengubah bunyi vokal aksara dasar /a/ yang mendahuluinya menjadi /o/.
-  = Pamingkal, berfungsi menambah bunyi /+ya/ pada aksara dasar yang dilekatinya, dan bisa disesuaikan dengan tanda vokalisasi pada aksara dasarnya.
-  = Pangwisad, berfungsi menambah konsonan /+h/ pada akhir aksara dasar.
-  = Pamaeh, berfungsi menghilangkan bunyi vokal pada aksara yang mendahuluinya [13]

2.6 Penulisan Aksara Sunda

Gambar 2.8 Contoh Penulisan Kata "Gaplok"

Dalam hal penulisan, pada aksara Sunda untuk menuliskan kata-kata atau kalimat yang tidak mempunyai konsonan rangkap atau gugus konsonan, maka penulisannya bisa dilakukan dengan sederhana, yaitu merangkaikan aksara ngalagena demi aksara ngalagena yang mewakili bunyi suara yang bersangkutan. Berbeda dengan hal tersebut, jika aksara Sunda digunakan untuk menuliskan kata-kata yang mempunyai gugus konsonan di tengah-tengah kata, maka penulisannya dapat dilakukan dengan menggunakan pamaéh. Setiap aksara ngalagena dapat mempunyai lebih dari satu Rarangkén [13]. Contoh penulisan aksara Sunda ditunjukkan pada Gambar 2.8. Pada tugas akhir ini, teks aksara Sunda mengikuti penulisan aksara Sunda baku sesuai aturan.

2.7 Capacity Ratio

Capacity ratio adalah kemampuan media yang dijadikan cover text dalam menyembunyikan informasi. Pada tugas akhir ini, *capacity ratio* digunakan untuk mengukur kemampuan teks aksara Sunda untuk menyimpan bit informasi mengacu pada metode *Feature Coding* yang digunakan.

$$CR = \frac{Text\ Capacity}{Text\ Size} \qquad (2.1)$$

Penghitungan *capacity ratio (bit/kilobyte)* ditunjukkan pada formula 2.1, *Text capacity (bit)* merupakan kapasitas informasi yang dapat disisipkan ke dalam teks. [4] Sementara itu, *text size (kilobyte)* adalah ukuran berkas. Pada tugas akhir ini, *text capacity* merupakan jumlah rarangkén, sedangkan *text size* merupakan ukuran berkas teks aksara Sunda yang menjadi *cover text*.

2.8 Java

Gambar 2.9 Logo Java

Java adalah bahasa pemrograman yang bersifat umum, konkuren, *class-based*, dan *object-oriented*. Java pertama kali dirilis pada tahun 1995 oleh Sun Microsystem dan didesain sebagai *production language*. [18] Sebagai bahasa tingkat tinggi, Java mempunyai *syntax* yang mudah dipelajari. Java tersedia pada berbagai platform seperti Linux, UNIX, Windows, dan Mac. Hasil kompilasi Java, bytecode, dapat dijalankan di berbagai platform selama sistem tersebut memiliki Java Runtime Environment (JRE). [19] Pada tugas akhir ini, Java merupakan bahasa pemrograman yang digunakan untuk mengembangkan sistem.

2.9 NetBeans

Gambar 2.10 Logo NetBeans

NetBeans adalah *Integrated Development Environment* (IDE) yang ditulis dengan bahasa pemrograman Java. NetBeans dikembangkan oleh Sun Microsystems. Dalam NetBeans, program atau aplikasi dikembangkan dari kumpulan komponen yang disebut *modules*. [20] NetBeans memungkinkan pembuatan aplikasi Java dengan antarmuka grafis. Pada tugas akhir ini, NetBeans digunakan sebagai perangkat pengembangan utama.

2.10 Apache PDFBox

Gambar 2.11 Logo Apache PDFBox

Apache PDFBox adalah *open source library* dalam Java untuk menangani format berkas PDF. Fitur dalam Apache PDFBox meliputi ekstraksi teks, *split & merge*, pengisian formulir, *preflight* (validasi), cetak, simpan sebagai gambar, pembuatan pdf, dan *signing*. [21] Pada tugas akhir ini, Apache PDFBox digunakan pada modul *embedding* sebagai *library* untuk menulis *cover text* aksara Sunda ke dalam bentuk PDF. Selain itu, *library* ini juga digunakan pada modul extracting untuk mengembalikan teks dari PDF yang telah dibuat.

2.11 FontForge

Gambar 2.12 Logo Fontforge

Fontforge adalah *font editor* untuk pembuatan dan modifikasi font dalam berbagai format standar. FontForge mendukung format Spline Font Database (SFD), TrueType (TTF), TrueType Collection (TTC), OpenType (OTF), dan format lainnya. [22] Pada tugas akhir ini, FontForge digunakan untuk modifikasi font Sundanese Unicode 2013.

2.12 Apache POI

Gambar 2.13 Logo Apache POI

Apache POI adalah *open source library* dalam Java untuk menangani format berkas yang digunakan Microsoft Office. Fitur utama Apache POI meliputi penulisan dan pembacaan berkas word, powerpoint dan excel [23]. Apache POI juga mendukung berkas Open Office XML. Pada tugas akhir ini, Apache POI digunakan untuk membaca berkas docx yang digunakan sebagai data masukan.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas analisis dan perancangan pada sistem yang akan dibangun. Sistem yang akan dibangun pada tugas akhir ini adalah steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding*. Sistem terbagi menjadi modul *embedding* dan *extracting*.

3.1 Data

Pada tugas akhir ini, data masukan yang digunakan dalam modul *embedding* dan *extracting* merupakan data yang berbeda. Data masukan akan diolah menjadi data keluaran melalui metode yang digunakan.

3.1.1 Data Modul Embedding

Data masukan dari modul embedding terdiri dari:

1. *Secret Message*

   Secret message merupakan pesan rahasia yang akan disisipkan ke dalam *cover text*. Pesan ini berupa huruf latin.
2. *Cover Text*

   Cover text merupakan teks yang digunakan untuk menyembunyikan *secret message*. Teks yang digunakan dalam tugas akhir ini berupa teks aksara Sunda dalam bentuk docx.
3. *Shifting Option*

   Shifting option merupakan opsi untuk pergeseran rarangkén. Terdapat dua opsi, yaitu menggeser semua rarangkén (*shift all*) dan menggeser rarangkén yang satu kelompok posisi (*shift group*)

Data keluaran dari modul embedding adalah *Stego Text*. *Stego text* merupakan teks yang telah disisipi *secret message*. Teks ini berupa teks aksara Sunda dalam bentuk PDF.

3.1.2 Data Modul Extracting

Data masukan dari modul *extracting* terdiri dari:

1. *Stego Text*

   Stego text merupakan teks yang telah disisipi *secret message*. Pada modul *extracting*, *stego text* akan diolah untuk didapatkan *secret message*.

2. *Shifting Option*

   Shifting option merupakan opsi dari pergeseran rarangkén yang digunakan. Opsi ini akan digunakan untuk menyesuaikan metode pencarian *secret message*.

Data keluaran dari modul *extracting* adalah *Secret Message*. *Secret message* merupakan pesan rahasia yang disisipkan ke dalam *stego text*. Pada modul *extracting*, *secret message* akan ditampilkan setelah *stego text* diolah oleh modul *extracting*.

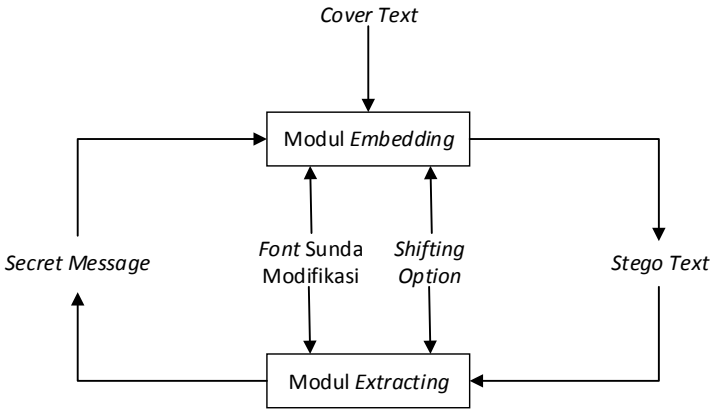
3.1.3 Data Font Modifikasi

Data *font* modifikasi adalah *font* Sundanese Unicode 2013 yang telah dilakukan modifikasi. *Font* Sundanese Unicode 2013 dimodifikasi menggunakan perangkat lunak FontForge. *Font* modifikasi terdiri dari blok yang normal dan blok yang telah digeser. Selain untuk menangani pergeseran, *font* modifikasi memiliki blok tambahan yang mengatur posisi rarangkén sejajar. Hal ini karena Apache PDFBox tidak mendukung OpenType *Layout* yang dimiliki oleh *font* Sundanese Unicode 2013.

Dalam sistem, *font* modifikasi akan digunakan pada modul *embedding* dan *extracting*. Instalasi dilakukan agar *font* ini dapat terdeteksi oleh sistem operasi dan sistem steganografi aksara Sunda.

3.2 Deskripsi Umum Sistem

Sistem steganografi teks pada aksara Sunda dalam tugas akhir ini terbagi atas modul *embedding* dan *extracting*. Sistem menggunakan pendekatan *Feature Coding* melalui pergeseran posisi rarangkén.



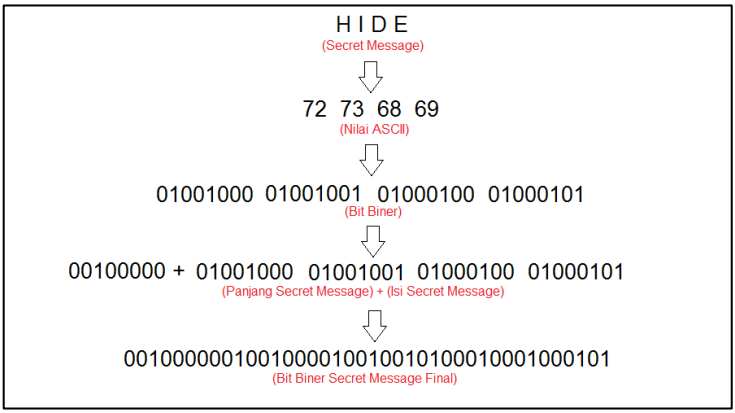
Gambar 3.1 Skema Umum Sistem

Pada modul embedding, sistem menampilkan antarmuka untuk pengisian data masukan *secret message*, *cover text*, dan *shifting option*. Modul *embedding* bertujuan untuk melakukan proses penyembunyian informasi. Hasil keluaran modul embedding adalah *stego text* dalam bentuk PDF.

Pada modul extracting, sistem menampilkan antarmuka untuk pengambilan berkas PDF dan pemilihan *shifting options*. Modul extracting melakukan ekstraksi *secret message* dari PDF. Hasil keluaran modul *extracting* adalah *secret message* yang dapat dilihat pada antarmuka program.

3.2.1 Deskripsi Modul Embedding

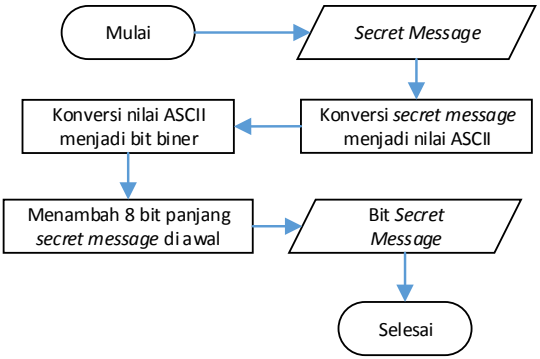
Modul *embedding* bertujuan untuk melakukan proses penyembunyian pesan rahasia. Modul embedding menerima data masukan berupa *secret message*, *shifting option*, dan *cover text.* Proses dalam modul embedding secara umum terbagi menjadi empat tahap yaitu konversi *secret message*, pembacaan *cover text*, pergeseran rarangkén, dan penyimpanan *stego text.*



Gambar 3.2 Contoh Konversi Bit *Secret Message*

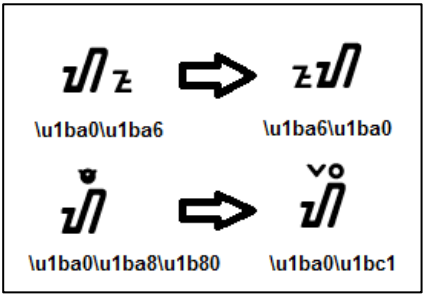
Proses konversi *secret message* dimulai dengan mengubah *secret message* ke dalam nilai ASCII untuk setiap huruf. Kemudian, setiap nilai ASCII tersebut diubah ke dalam bentuk bit biner. Satu nilai ASCII direpresentasikan dengan 8 bit biner. Bit biner dari *secret message* ditambahkan 8 bit atau lebih di awal.

Pada Gambar 3.2. ditunjukkan contoh konversi kata “HIDE” yang merupakan *secret message*. Setiap karakter dari kata “HIDE” diubah ke dalam nilai ASCII yang menghasilkan “72 73 68 69”. Masing-masing empat nilai tersebut diubah ke dalam bentuk bit biner. Pada contoh ini, jumlah bit panjang *secret message* yang ditambahkan di awal berjumlah 8 bit. Diagram alir proses konversi *secret message* ditunjukkan pada Gambar 3.3.



Gambar 3.3 Diagram Alir Konversi *Secret Message*

Selanjutnya, pada proses pembacaan *cover text*, sistem membaca cover text dalam bentuk docx menggunakan Apache POI. Cover text yang didapatkan lalu diubah menjadi nilai Unicode. Kemudian, nilai Unicode melalui proses normalisasi untuk menghasilkan posisi rarangkén yang sesuai. Hal ini dilakukan karena Apache PDFBox tidak mendukung OpenType *Layout* yang terdapat pada *font* Sundanese Unicode 2013.



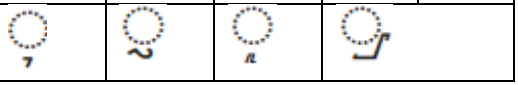
Gambar 3.4 Contoh Normalisasi *Cover Text*

Normalisasi yang dilakukan mencakup perbaikan posisi rarangkén paneleng dan rarangkén yang bertumpuk seperti ditunjukkan pada Gambar 3.4. Secara umum rarangkén paneleng yang seharusnya berada di depan menjadi berada di belakang

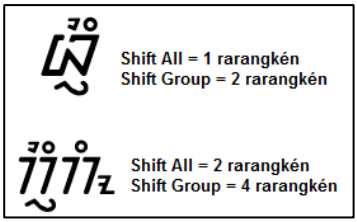
ketika dijalankan pada lingkungan Java. Pada kasus ini, sistem menukar nilai Unicode menjadi di depan. Sementara itu, aksara ngalagena yang memiliki dua rarangkén pada posisi yang sama yang seharusnya terlihat berpasangan, tampak bertumpuk satu sama lain. Untuk menangani masalah ini, sistem mengganti nilai Unicode dari dua rarangkén tersebut dengan satu nilai Unicode baru.

Sistem menggunakan nilai Unicode yang telah dinormalisasi untuk menghitung jumlah rarangkén. Penghitungan ini dipengaruhi oleh *shifting option* yang dipilih sebelumnya. Dalam *shift all*, satu aksara ngalagena yang memiliki satu atau banyak rarangkén hanya dihitung sebagai satu rarangkén. Dalam *shift group*, satu aksara ngalagena yang memiliki banyak rarangkén, dihitung berdasarkan jumlah kelompok posisi rarangkén yang ada.

Tabel 3.1 Kelompok Posisi Rarangkén

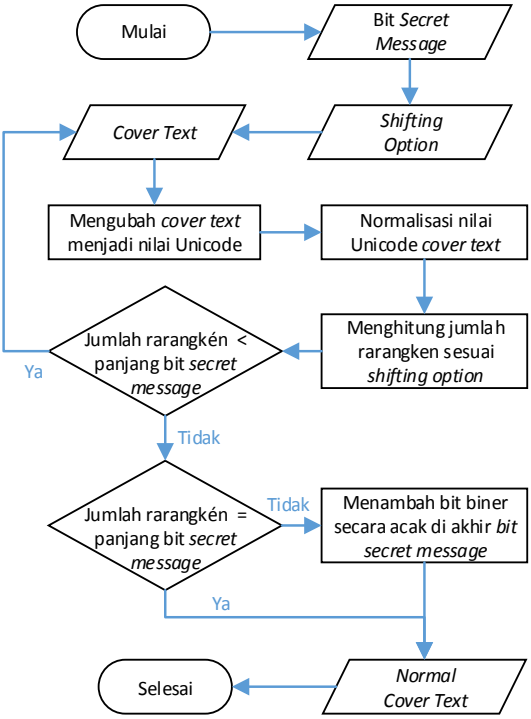
Posisi	**Rarangkén**
Depan	
Belakang	
Atas	
Bawah	

Pembagian kelompok posisi rarangkén ditunjukkan pada Tabel 3.1, Pembagian ini hampir sama dengan pengelompokan posisi rarangkén standar. Perbedaan terletak pada pemisahan kategori rarangkén yang sejajar dengan aksara ngalagena, dibagi menjadi depan dan belakang. Selain itu, rarangkén pamingkal dikelompokan menjadi bagian bawah karena mempunyai bagian seperti ekor yang menjulur ke bawah.



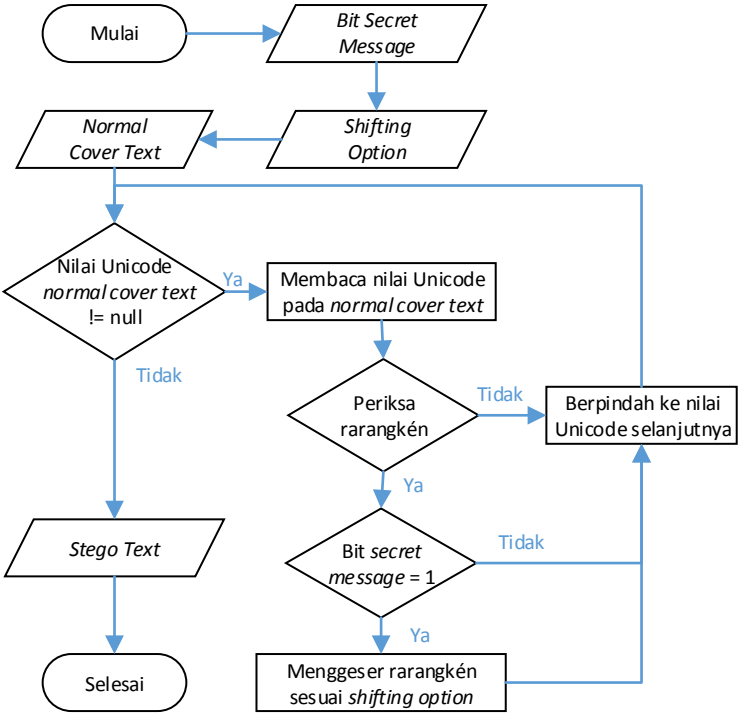
Gambar 3.5 Contoh Penghitungan Rarangkén

Contoh penghitungan rarangkén ditunjukkan pada Gambar 3.5. Aksara ngalagena yang memiliki dua rarangkén atas dan satu rarangkén bawah akan dihitung sebagai 2 rarangkén jika menggunakan *shift group*, karena terdapat 2 kelompok posisi rarangkén, yaitu atas dan bawah.



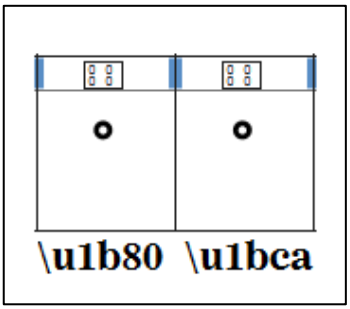
Gambar 3.6 Diagram Alir Pembacaan *Cover Text*

Setelah mendapatkan jumlah rarangkén, sistem membandingkannya dengan panjang bit *secret message*. Jika jumlah rarangkén lebih dari panjang bit *secret message*, maka sistem menambah bit biner acak di belakang bit *secret message* sesuai selisih jumlah rarangkén dan panjang bit *secret message.* Jika jumlah rarangkén kurang dari panjang bit *secret message*, maka sistem kembali ke antarmuka pengisian masukan. Jika jumlah rarangkén sama dengan panjang bit *secret message*, maka proses pergeseran rarangkén dapat dilakukan. Diagram alir proses pembacaan cover text secara umum ditunjukkan pada Gambar 3.6.



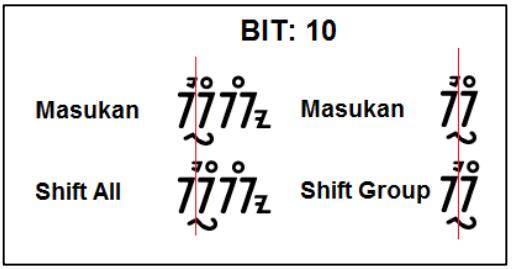
Gambar 3.7 Diagram Alir Pergeseran Rarangkén

Selanjutnya, proses pergeseran rarangkén diatur berdasarkan bit *secret message* dan *shifting option*. Sistem memeriksa setiap nilai Unicode yang ada. Jika termasuk rarangkén maka dilakukan pengecekan bit *secret message* yang ada. Jika bit secret message bernilai "1", maka rarangkén mengalami pergeseran, tetapi jika bit *secret message* bernilai "0", maka rarangkén tidak mengalami pergeseran. Secara umum diagram alir untuk proses ini ditunjukkan pada Gambar 3.7.



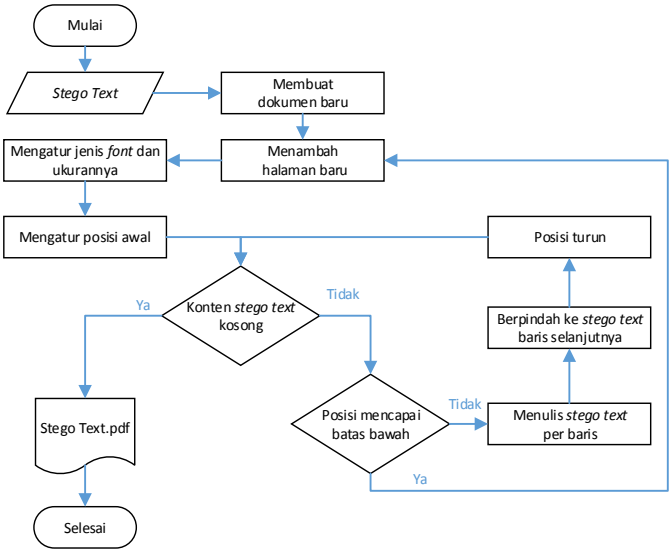
Gambar 3.8 Blok Panolong Asal dan Modifikasi

Proses pergeseran rarangkén dilakukan dengan mengganti huruf yang mewakili rarangkén normal pada blok Unicode asal dengan huruf yang mewakili rarangkén modifikasi pada blok Unicode baru. Pergeseran yang dilakukan mengacu pada masukan *shifting option*.



Gambar 3.9 Perbedaan *Shifting Option*

Sistem menggeser atau tidak semua rarangkén dalam satu aksara ngalagena untuk satu bit jika opsi yang dipilih adalah *shift all*. Pada opsi *shift group*, sistem menggeser atau tidak satu kelompok posisi rarangkén dalam satu huruf untuk satu bit. Contoh pergeseran dan *shifting option* yang dipilih ditunjukkan pada gambar 3.9.

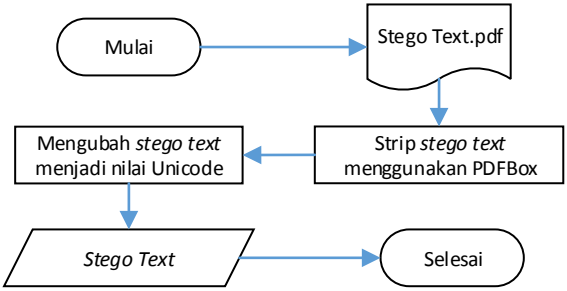


Gambar 3.10 Diagram Alir Penyimpanan *Stego Text*

Setelah mendapatkan *stego text*, sistem melakukan proses penyimpanan *stego text*. Proses yang menggunakan Apache PDFBox ini dimulai dengan pembuatan dokumen baru. Setelah itu, sistem menambahkan halaman pada dokumen tersebut. Sebelum memulai menulis *stego text*, sistem menentukan jenis *font* dan ukurannya. Sistem juga menentukan posisi awal yang berada di atas. Jenis *font* yang digunakan adalah *font* Sundanese Unicode 2013 hasil modifikasi. Sistem mulai menulis *stego text* dari posisi awal. Ketika *stego text* sudah mencapai batas bawah posisi, maka sistem membuat halaman baru, mengatur ulang *font* dan memulai kembali menulis *stego text* dari posisi atas sampai selesai.

3.2.2 Deskripsi Modul Extracting

Modul *extracting* bertujuan untuk mengembalikan *secret message* dari *stego text* yang diberikan. Modul *extracting* menerima data masukan berupa *shifting option* dan *stego text* dalam bentuk PDF. Secara umum terdapat 3 proses dalam modul *extracting* yaitu pembacaan *stego* text, penghitungan panjang *secret message*, dan ekstraksi *secret message*.

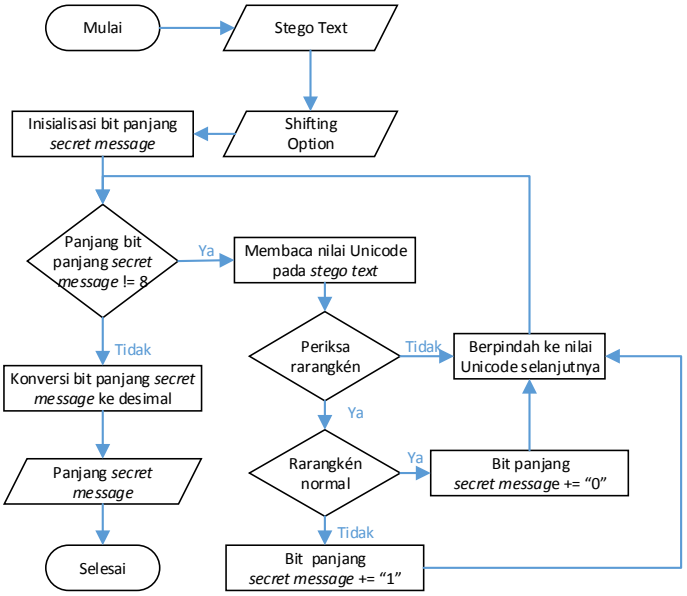


Gambar 3.11 Diagram Alir Pembacaan *Stego Text*

Proses pembacaan *stego text* dimulai dengan membaca data berkas PDF yang berisi *stego text*. Pembacaan ini dilakukan menggunakan bantuan text stripper dari Apache PDFBox. Hasil pembacaan data berupa string aksara Sunda. Hasil ini kemudian diubah menjadi nilai Unicode untuk dilakukan penghitungan panjang secret message.

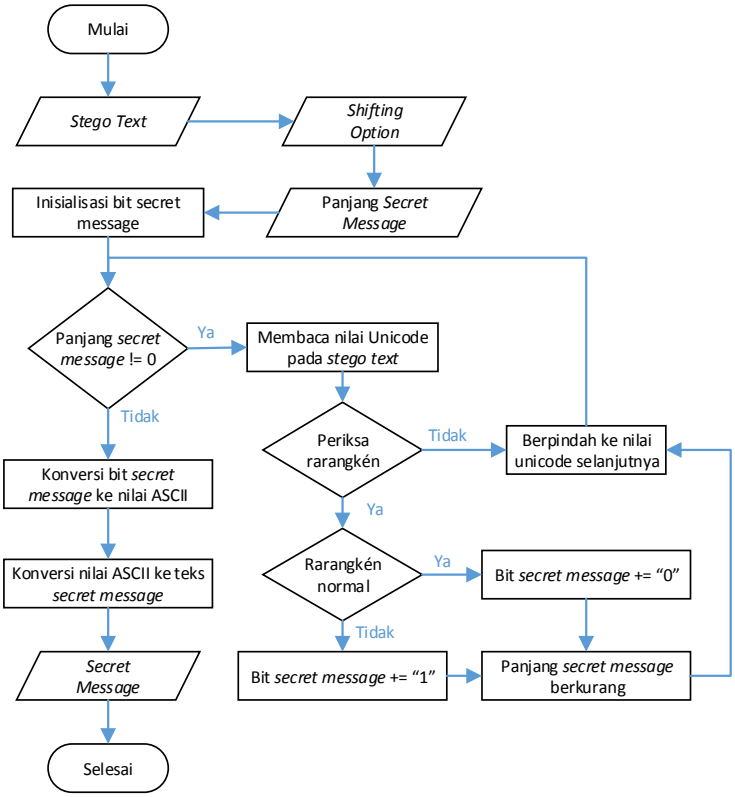
Selanjutnya, proses penghitungan panjang *secret message* dilakukan dengan memeriksa setiap rarangkén. Pemeriksaan dipengaruhi oleh *shifting option*. Setiap rarangkén diperiksa apakah mengalami pergeseran atau tidak. Rarangkén yang mengalami pergeseran menempati blok Unicode modifikasi ketika diperiksa. Jika suatu rarangkén *stego text* mengalami pergeseran, maka didapatkan bit panjang secret message bernilai "1". Sebaliknya, jika suatu rarangkén tidak mengalami pergeseran, maka bit yang didapatkan bernilai "0". Proses penghitungan panjang berhenti ketika panjang bit panjang *secret message* yang didapatkan berjumlah sesuai dengan jumlah bit yang telah

ditentukan sebelumnya. Dalam hal ini, bit berjumlah 8 bit atau lebih. Jumlah bit yang ditentukan sama pada modul *embedding* maupun *extracting*. Bit panjang *secret message* tersebut dikonversi ke dalam bentuk desimal. Diagram alir untuk panjang bit panjang *secret message* berjumlah 8 ditunjukkan pada gambar 3.12.



Gambar 3.12 Diagram Alir Penghitungan Panjang *Secret Message*

Proses ekstraksi *secret message* dilakukan dengan melanjutkan pemeriksaan setiap rarangkén setelah proses penghitungan panjang *secret message*. Setelah didapatkan bit sejumlah panjang *secret message* yang didapatkan pada proses sebelumnya, pemeriksaan rarangkén berhenti. Bit *secret message* yang didapatkan pada proses ekstraksi dikonversi ke dalam nilai ASCII untuk setiap 8 bit. Setelah itu, nilai ASCII akan dikembalikan ke dalam bentuk teks *secret message*. Teks tersebut ditampilkan dalam antarmuka program . Diagram Alir proses ini ditunjukkan pada Gambar 3.13.

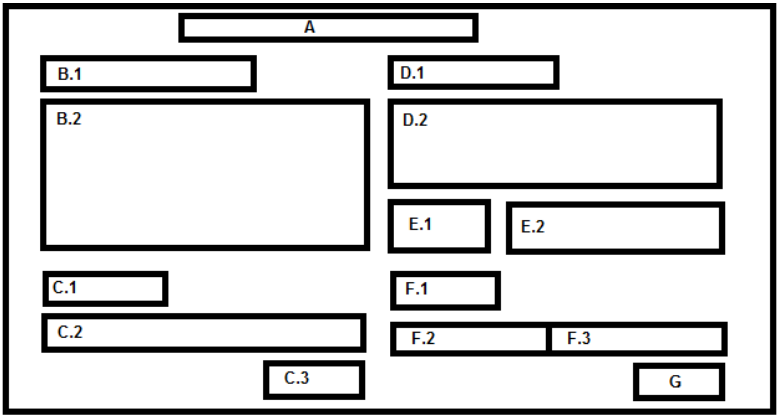


Gambar 3.13 Diagram Alir Ekstraksi Secret Message

3.3 Perancangan Antarmuka

Antarmuka digunakan untuk memberikan kemudahan kepada pengguna sistem. Pada tugas akhir ini, antarmuka sistem terbagi menjadi antarmuka modul *embedding* dan antarmuka modul *extracting*. Antarmuka sistem akan dibangun menggunakan JFrame *class* dalam NetBeans IDE 8.1, Rincian desain antarmuka modul dapat dilihat pada sub-bab di bawah.

3.3.1 Antarmuka Modul Embedding



Gambar 3.14 Perancangan Antarmuka Modul *Embedding*

Desain antarmuka modul *embedding* ditunjukkan pada Gambar 3.14. Rincian komponen dalam desain tersebut antara lain:

A. Bagian A merupakan label untuk menunjukkan nama sistem.

B. Bagian B merupakan bagian untuk memasukkan data *secret message*. B.1 adalah label yang menandakan *secret message*. Kolom masukkan untuk *secret message* berada pada B.2.

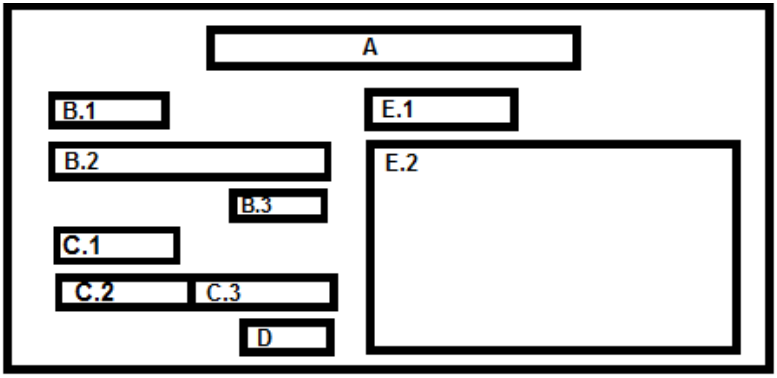
C. Bagian C merupakan bagian untuk memasukkan data *cover text*. Label penanda ditunjukkan pada bagian C.1, Sementara itu, bagian C.2 menandakan kolom yang berisi *path* dari *cover text*. Kolom ini tidak dapat diubah nilainya secara manual dan hanya akan berubah nilainya ketika bagian C.3 ditekan. Bagian C.3 merupakan *tombol* yang memunculkan *dialog box* pemilihan berkas *cover text.*

D. Bagian D merupakan bagian untuk melihat hasil konversi dari *secret message* menjadi bit biner. Bagian D.1 menandakan label *binary value*. Pada D.2, deret biner

terlihat sebagai hasil konversi. D.2 tidak dapat menerima masukan dan hanya memunculkan hasil ketika tombol pada bagian G ditekan.

E. Bagian E merupakan bagian untuk melihat hasil penghitungan rarangkén. E.1 menandakan label total rarangkén. Hasil penghitungan akan muncul pada bagian E.2. Bagian E.2 tidak dapat menerima masukan dan hanya memunculkan hasil setelah tombol pada bagian G ditekan
F. Bagian F merupakan bagian untuk memilih *shifting option*. Bagian F.1 menandakan label *shifting option*. F.2 dan F.3 berturut-turut merupakan pilihan *shifting option* yang tersedia, yaitu *shift all* dan *shift group*. Pengguna hanya dapat memilih salah satu diantara ke dua pilihan tersebut.
G. Bagian G merupakan bagian untuk memulai proses *embedding*. Bagian G direpresentasikan dengan tombol. Setelah tombol G ditekan, maka hasil pada D.2 dan E.2 akan terlihat.

3.3.2 Antarmuka Modul Extracting



Gambar 3.15 Perancangan Antarmuka Modul Extracting

Desain antarmuka modul *extracting* ditunjukkan pada Gambar 3.15. Rincian komponen dalam desain tersebut antara lain:

A. Bagian A merupakan label untuk menunjukkan nama sistem.

B. Bagian B merupakan bagian untuk memasukkan data *stego text*. B.1 menandakan label untuk *stego text.* B.2 merupakan *path* dari stego text yang dipilih. Pemilihan stego text dilakukan ketika bagian B.3 yang merupakan tombol ditekan. Ketika B.3 ditekan, maka akan muncul *dialog box* untuk memilih berkas *stego text.*

C. Bagian C merupakan bagian untuk *shifting option*. C.1 adalah label dari *shifting option*. Kemudian C.2 mewakili opsi *shift all* dan C.3 untuk opsi *shift group*. Pengguna hanya dapat memilih salah satu dari C.2 dan C.3

D. Bagian D merupakan bagian untuk melakukan proses pada modul *extracting*. Bagian ini merupakan tombol dan akan memunculkan hasil ekstraksi *secret message* pada E.2.

E. Bagian E merupakan bagian untuk menampilkan hasil dari proses pada modul *extracting*. E.1 adalah label *secret message*. Hasil *secret message* ditampilkan pada E.2 setelah tombol pada bagian D ditekan.

BAB IV
IMPLEMENTASI

Bab ini menguraikan tentang implementasi sistem steganografi teks pada aksara Sunda. Implementasi dikerjakan berdasarkan perancangan pada bab sebelumnya. Pada bab ini, implementasi meliputi modifikasi *font*, modul *embedding* dan modul *extracting*.

4.1 Lingkungan Implementasi

Implementasi sistem steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding* menggunakan perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1,

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat	Jenis	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i5-4200U CPU @ 1,60GHz
	Memori	4GB 1600 MHz DDR3
Perangkat Lunak	Sistem Operasi	Windows 8.1
	Bahasa Pemrograman	Java
	Perangkat Pengembang	NetBeans IDE 8.1
	Font Editor	FontForge 20:55

4.2 Implementasi Modifikasi Font

Modifikasi *font* dilakukan menggunakan FontForge 20:55. Modifikasi dilakukan pada *font* Sundanese Unicode 2013. Rarangkén yang mempunyai kombinasi berpasangan dalam satu posisi akan digabungkan dan disimpan ke dalam nilai Unicode baru.

Tabel 4.2 Implementasi Modifikasi Rarangkén Tunggal

BLOK UNICODE NORMAL												
\u1b80	\u1b81	\u1b82	\u1ba1	\u1ba2	\u1ba3	\u1ba4	\u1ba5	\u1ba6	\u1ba7	\u1ba8	\u1ba9	\u1baa
◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
BLOK UNICODE GESER												
\u1bca	\u1bcb	\u1bcc	\u1bcd	\u1bce	\u1bcf	\u1bd0	\u1bd1	\u1bd2	\u1bd3	\u1bd4	\u1bd5	\u1bd6
◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌

Setiap rarangkén pada font tersebut, baik tunggal maupun tambahan kombinasi, dibuat replikanya yang disimpan pada blok Unicode baru. Replika tersebut bergeser dari posisi seharusnya. Hasil implementasi modifikasi ditunjukkan pada Tabel 4.2 dan Tabel 4.3.

Tabel 4.3 Implementasi Modifikasi Rarangkén Kombinasi

BLOK UNICODE NORMAL								
\u1bc0	\u1bc1	\u1bc2	\u1bc3	\u1bc4	\u1bc6	\u1bc7	\u1bc8	\u1bc9
◌	◌	◌	◌	◌	◌	◌	◌	◌
BLOK UNICODE GESER								
\u1bd8	\u1bd9	\u1bda	\u1bdb	\u1bdc	\u1bdd	\u1bde	\u1bdf	\u1be0
◌	◌	◌	◌	◌	◌	◌	◌	◌

4.3 Implementasi Modul Embedding

Implementasi modul *embedding* secara umum terbagi menjadi 4 tahapan proses meliputi: konversi secret message, pembacaan *cover text*, pergeseran rarangkén, dan penyimpanan *stego text*. Keterangan implementasi untuk setiap tahapan proses terdapat pada masing-masing sub-bab.

4.3.1 Implementasi Konversi Secret Message

Proses konversi secret message menggunakan data masukan *secret message*. *Pseudocode* proses ini ditunjukkan pada Gambar 4.1, Sistem menyimpan hasil konversi *secret message* pada variabel binary_message (*string*). Untuk melakukan konversi, sistem memanggil fungsi getBinaryValue seperti ditunjukkan pada baris 3.

```
1  READ secret_message

2  binary_message <-- ""
3  binary_message <-- CALL getBinaryValue with
   secret_message RETURNING binary_message

4  OUT binary_message
```

Gambar 4.1 Implementasi Konversi Secret Message

Rincian fungsi getBinaryValue dtunjukkan pada Gambar 4.2. Fungsi ini memiliki parameter secret_message (string). Parameter secret_message diubah ke dalam nilai ASCII dengan mengambil *byte*-nya dan disimpan pada variabel secret_byte (*array of byte*). Untuk mendapatkan nilai biner per byte, sistem melakukan perulangan sebanyak panjang secret_byte. Setiap perulangan sistem menambahkan nilai binary_message (*string*) yang merupakan hasil perubahan secret_byte menjadi nilai *string* biner menggunakan toBinaryString. Setelah selesai melakukan perulangan, binary_message akan ditambahkan bit biner panjang binary_message tersebut. Fungsi mengembalikan nilai binary_message akhir.

```
1   FUNCTION getBinaryValue(secret_message):

2   secret_byte <-- []
3   binary_message <-- ""
4   secret byte <-- secret_message.getBytes()

5   FOR i in range of secret_byte.length
6         binary_message   <--   binary_message   +
    toBinaryString(secret_byte[i])
7         increment i

8   binary_message<--
    toBinaryString(binary_message.length)         +
    binary_message

9   RETURN binary_message
10  ENDFUNCTION
```

Gambar 4.2 Implementasi Fungsi getBinaryValue

4.3.2 Implementasi Pembacaan Cover Text

Proses pembacaan *cover text* menggunakan data binary_message (*string*) dan option (*int*) yang merupakan *shifting option.* Secara umum proses ditunjukkan pada Gambar 4.3. Pada proses ini, sistem memanggil fungsi readCoverText untuk membaca dan normalisasi cover_text. Hasil fungsi readCoverText disimpan dalam unicode_cover (*array of string*). Selanjutnya, dilakukan proses pembacaan dan penghitungan rarangkén pada fungsi readRarangkén. Hasil fungsi readRarangkén memberi nilai baru pada binary_message.

```
1   READ binary_message, option

2   unicode_cover <-- []
3   unicode_cover <-- CALL readCoverText RETURNING
    unicode_cover
4   binary_message  <--  CALL  readRarangkén  with
    unicode_cover, option, binary_message RETURNING
    binary_message

5   OUT binary_message, unicode_cover, option
```

Gambar 4.3 Implementasi Pembacaan Cover Text

Rincian fungsi readCoverText ditunjukkan pada Gambar 4.4. Fungsi memanggil openCoverText untuk mendapatkan nilai cover_text (*string*) seperti ditunjukkan pada baris 5. Fungsi mendapatkan nilai location dari antarmuka pengisian. Setelah mendapatkan cover_text, sistem mengubahnya menjadi nilai Unicode dengan fungsi getUnicodeValue. Hasil fungsi getUnicodeValue disimpan dalam variabel unicode_cover (array of string). Selanjutnya, sistem memanggil fungsi normalizeRarangkén untuk melakukan normalisasi rarangkén pada unicode_cover. Hasil fungsi normalizeRarangkén memberi nilai baru pada unicode_cover. Fungsi readCoverText secara keseluruhan mengembalikan nilai unicode_cover akhir.

```
1   FUNCTION readCoverText()

2   READ location
3   cover_text <-- ""
4   unicode_cover <-- []

5   cover_text <-- CALL openCoverText with location
    RETURNING cover_text
6   unicode_cover <-- CALL getUnicodeValue with
    cover_text RETURNING unicode_cover
7   unicode_cover <-- CALL normalizeRarangkén with
    unicode_cover RETURNING normal_cover

8   RETURN unicode_cover

9   ENDFUCTION
```

Gambar 4.4 Implementasi Fungsi readCoverText

Rincian fungsi readRarangkén ditunjukkan pada Gambar 4.5. Fungsi memanggil countRarangkén untuk mendapatkan jumlah rarangkén pada *cover text*. Hasil countRarangkén disimpan ke dalam variabel total_rarangkén (int). Jika total_rarangkén kurang dari panjang binary_message, maka sistem menampilkan pesan peringatan dan sistem berhenti. Jika total_rarangkén lebih dari panjang binary_message, maka sistem menambahkan bilang biner acak di belakang binary_message sehingga panjang

binary_message dan total_rarangkén menjadi sama. Penambahan tersebut menggunakan addRandomBinary. Fungsi readRarangkén secara umum mengembalikan binary_message.

```
1   FUNCTION readRarangkén(unicode_cover, option,
    binary_message)

2   total_rarangkén <-- 0

3   total_rarangkén <-- CALL countRarangkén with
    unicode_cover, option RETURNING total_rarangkén

4   IF total_rarangkén < binary_message.length THEN
5       show message box "your cover text is
    insufficient"
6       EXIT
7   ELSE IF total_rarangkén > binary_message.length
    THEN
8       binary_message <-- binary_message + CALL
    addRandomBinary with total_rarangkén,
    binary_message RETURNING binary_random

9   RETURN binary_message

10  ENDFUNCTION
```

Gambar 4.5 Implementasi Fungsi readRarangkén

4.3.2.1 Implementasi Fungsi openCoverText

Fungsi openCoverText bertujuan untuk mendapatkan *cover text* dari data masukan berupa docx. Fungsi ini menggunakan bantuan *library* Apache POI. Ketika fungsi dipanggil, fungsi membuat XWPFDocument dari path yang ada. Selanjutnya, fungsi mengekstrak konten dari docx. Konten yang diekstrak diambil teksnya dan disimpan dalam variable cover_text. Fungsi mengembalikan cover_text (*string*). *Pseudocode* fungsi ditunjukkan pada Gambar 4.6

```
1   FUNCTION openCoverText(path)

2   cover_text <-- ""
3   docx <-- null
4   content <-- null

5   docx <-- new XWPFDocument from path
6   content <-- extract docx with XWPFWordExtractor
7   cover_text <-- content.getText()

8   RETURN cover_text
9   ENDFUNCTION
```

Gambar 4.6 Implementasi Fungsi openCoverText

4.3.2.2 Implementasi Fungsi getUnicodeValue

Fungsi getUnicodeValue bertujuan untuk mendapatkan nilai unicode dari cover_text. Ketika fungsi dipanggil, sistem mengubah cover_text menjadi unicode_char (*array of char*). Selanjutnya, sistem melakukan perulangan pada setiap char dalam unicode_char. Sistem mengubah char menjadi hex string dan menyimpannya dalam unicode_cover. Akumulasi unicode_cover diproses menggunakan stripMessage untuk menghasilkan unicode_stripped (*array of string*). Fungsi getUnicodeValue mengembalikan nilai unicode_cover akhir. *Pseudocode* fungsi ditunjukkan pada Gambar 4.7.

```
1   FUNCTION getUnicodeValue(cover_text)
2   unicode_cover <-- ""
3   unicode_char[] <-- toCharArray(cover_text)
4   unicode_stripped <-- []

5   FOR c in unicode_char
6         unicode_cover <-- unicode_cover + "\\u" +
    toHexString(c)

7   unicode_stripped <-- CALL stripMessage with
    unicode_cover RETURNING stripped

8   RETURN unicode_stripped
9   ENDFUNCTION
```

Gambar 4.7 Implementasi Fungsi getUnicodeValue

4.3.2.3 Implementasi Fungsi stripMessage

Fungsi stripMessage bertujuan untuk membagi *string* Unicode dari *cover text*, menjadi *array of string* agar mempermudah pemeriksaan rarangkén. Dalam melakukan pembagian, sistem menghilangkan spasi dengan *split*. Selanjutnya, sistem menghilangkan "\\" dengan *replace*. Hasil tersebut dilakukan *split* kembali berdasarkan "u" sehingga terbentuk stripped (*array of string*). Hasil stripped ini berupa nilai heksadesimal dari masing-masing Unicode. Fungsi mengembalikan stripped akhir. *Pseudocode* fungsi ditunjukkan pada Gambar 4.8

```
1  FUNCTION stripMessage(message)
2  strip_message <-- ""
3  stripped <-- []

4  strip_message <-- message.split(" ")[0]
5  strip_message<-- strip_message.replace("\\","")
6  stripped <-- strip_message.split("u")

7  return stripped
8  ENDFUNCTION
```

Gambar 4.8 Implementasi Fungsi stripMessage

4.3.2.4 Implementasi Fungsi normalizeRarangkén

Fungsi normalizeRarangkén bertujuan untuk melakukan normalisasi pada rarangkén. *Pseudocode* fungsi normalizeRarangkén ditunjukkan pada Gambar 4.9. Sistem melakukan perulangan selama panjang cover_stripped. Pada kondisi IF di baris 5, sistem mem-*filter* nilai unicode selain dari aksara Sunda dan aksara Sunda yang terakhir. Pada kondisi ELSE IF di baris 7, sistem melakukan normalisasi untuk posisi rarangkén panghulu yang berpasangan. Pada kondisi ELSE IF di baris 17, sistem melakukan normalisasi untuk posisi rarangkén pamepet yang berpasangan. Pada Kondisi ELSE IF di baris 27, sistem melakukan normalisasi untuk posisi rarangkén paneuleung yang berpasangan. Pada Kondisi ELSE IF di baris 37, sistem melakukan normalisasi untuk posisi rarangkén panyuku yang berpasangan.

Normalisasi pada rarangkén paneling, ditunjukkan pada kondisi di baris 58 dan 62.

```
1   FUNCTION normalizeRarangkén(cover_stripped)
3   normal_cover <-- ""
3   normal_stripped <-- []

4   FOR i in range of cover_stripped.length
5     IF cover_stripped startsWith "00" or i+1 > cover_stripped.length THEN
6       normal_cover <-- normal_cover + "\\u" + cover_stripped[i]
7     ELSE IF cover_stripped[i] is "1ba4" THEN
8       Case based on cover_stripped[i+1]
9         case "1b80"
10          normal_cover <-- normal_cover + "\\u1bc0"
11          increment i
12        case "1b81"
13          normal_cover <-- normal_cover + "\\u1bc3"
14          increment i
15        default
16          normal_cover <-- normal_cover + "\\u1ba4"
17    ELSE IF cover_stripped[i] is "1ba8" THEN
18      Case based on cover_stripped[i+1]
19        case "1b80"
20          normal_cover <-- normal_cover + "\\u1bc1"
21          increment i
22        case "1b81"
23          normal_cover <-- normal_cover + "\\u1bc4"
24          increment i
25        default
26          normal_cover <-- normal_cover + "\\u1ba8"
27    ELSE IF cover_stripped[i] is "1ba9" THEN
28      Case based on cover_stripped[i+1]
29        case "1b80"
30          normal_cover <-- normal_cover + "\\u1bc2"
31          increment i
32        case "1b81"
33          normal_cover <-- normal_cover + "\\u1bc5"
34          increment i
35        default
36          normal_cover <-- normal_cover + "\\u1ba9"
37    ELSE IF cover_stripped[i+1] is "1ba5" THEN
38      temp <--- "\\u1ba5"
```

```
39      Case based on cover_stripped[i]
40        case "1ba1"
41          temp = "\\u1bc8"
42          increment i
43        case "1ba2"
44          temp = "\\u1bc6"
45          increment I
46        case "1ba3"
47          temp = "\\u1bc7"
48          increment i
49        default
50        IF temp is "\\u1ba5" THEN
51            normal_cover <-- normal_cover + "\\u"
    + cover_stripped[i] + temp
52            increment i
53         ELSE   IF   CALL   checkRarangkén   with
    cover_stripped[i+1] RETURNING status is true THEN
54            normal_cover <-- normal_cover + "\\u"
    + cover_stripped[i+1] + temp
55            increment i
56         ELSE
57             normal_cover <--- normal_cover + temp
58    ELSE IF cover_stripped[i+1] is "1ba6" THEN
59      normal_cover <-- normal_cover + "\\u1ba6"
60      normal_cover  <--  normal_cover  +  "\\u"  +
    cover_stripped[i]
61          increment i
62     ELSE  IF  i+2  <  cover_stripped.length  and
    cover_stripped[i+2] is "1ba6" THEN
63       normal_cover <-- normal_cover + "\\u1ba6"
64       normal_cover  <--  normal_cover  +  "\\u"  +
    cover_stripped[i]
65       normal_cover  <--  normal_cover  +  "\\u"  +
    cover_stripped[i+1]
66         i <-- i+2;
67     ELSE
68       normal_cover  <--  normal_cover  +  "\\u"  +
    cover_stripped[i]

69  normal_stripped <-- stripMessage(normal_cover)
70  return normal_cover
74  ENDFUNCTION
```

Gambar 4.9 Implementasi Fungsi normalizeRarangkén

4.3.2.5 Implementasi Fungsi checkRarangkén

Fungsi checkRarangkén bertujuan untuk memeriksa apakah suatu nilai Unicode termasuk rarangkén atau tidak. Fungsi ini mengubah nilai Unicode menjadi nilai integer heksadesimal melalui parseInt. Selanjutnya, sistem mengembalikan nilai Boolean sesuai kondisi yang ada. Jika hex_val (*int*) berada pada rentang 7040-7042, 7073-7082 atau 7104-7112, maka sistem mengembalikan nilai *true*. Jika hex_val berada di luar tiga rentang tersebut, maka fungsi mengembalikan nilai *false*. *Pseudocode* fungsi ditunjukkan pada Gambar 4.10.

```
1   FUNCTION checkRarangkén(stripped)

2   hex_val <-- 0
3   hex_val <-- parseInt(stripped) to HEX
4   return (hex_val >= 7040 and hex_val <= 7042) or
    (hex_val >= 7073 and hex_val <= 7082) or (hex_val
    >=7104 and hex_val <=7112)

5   ENDFUNCTION
```

Gambar 4.10 Implementasi Fungsi checkRarangkén

4.3.2.6 Implementasi Fungsi countRarangkén

Fungsi countRarangkén bertujuan untuk menghitung jumlah rarangkén dalam *cover text*. Sistem melakukan perulangan sebanyak panjang cover_stripped. Jika sistem menemukan rarangkén dengan memanggil checkRarangkén, maka count (*int*) yang merupakan variabel penghitung jumlah rarangkén bertambah. Jika option yang dipilih adalah 1 (*shift all*), maka sistem melewati sisa rarangkén yang terdapat pada suatu aksara ngalagena tanpa menghitungnya ke dalam count. Sebaliknya, jika option adalah 0 (*shift group*), maka sistem hanya melewati rarangkén dalam satu kelompok posisi. *Pseudocode* fungsi ditunjukkan pada Gambar 4.11.

```
1   FUNCTION countRarangkén(cover_stripped, option)
2   count <-- 0
3   FOR i in range of cover_stripped.length
4       IF CALL checkRarangkén with cover_stripped[i]
    RETURNING status is true THEN
5           increment count
6           IF option is 1
7               IF cover_stripped[i] is 1ba6 THEN
8                   increment i
9               i <-- i + CALL countShiftAll with i,
    cover_stripped RETURNING counter
10              ELSE
11                  IF cover_stripped[i] is "1ba7" and
    cover_stripped[i+1] is "1b82" THEN
12                      increment i
13  RETURN count
14  ENDFUNCTION
```

Gambar 4.11 Implementasi Fungsi countRarangkén

4.3.2.7 Implementasi Fungsi countShiftAll

Fungsi countShiftAll bertujuan untuk menghitung jumlah rarangkén yang harus dilewati dalam suatu aksara ngalagena. Fungsi ini dipanggil ketika option yang dipilih adalah 1 (*shift all*). Sistem melakukan perulangan selama cover_stripped yang selanjutnya adalah rarangkén. Jika tidak, maka sistem berhenti dan mengembalikan nilai penghitungan rarangkén. *Pseudocode* fungsi ditunjukkan pada Gambar 4.12.

```
1   FUNCTION countShiftAll(index, cover_stripped)

2   counter <-- 0
3   WHILE index+1 < cover_stripped.length and CALL
    checkRarangkén   with   cover_stripped[index+1]
    RETURNING status is true
4   DO
5      IF cover_stripped[index+1] is "1ba6"
6          break
7      increment counter
8      increment index
9   RETURN counter
10  ENDFUNCTION
```

Gambar 4.12 Implementasi Fungsi countRarangkén

4.3.3 Implementasi Pergeseran Rarangkén

Proses pergeseran rarangkén menggunakan data unicode_cover, binary_message, dan option. Pergeseran rarangkén dilakukan dengan memanggil fungsi shiftRarangkén yang mengembalikan nilai Unicode yang telah digeser. Hasil tersebut disimpan dalam variable unicode_stego (*array of string*). *Pseudocode* proses secara umum ditunjukkan pada Gambar 4.13.

```
    READ unicode_cover, binary_message, option

    unicode_stego <-- []
    unicode_stego  <--  CALL  shiftRarangkén  with
    unicode_cover, binary_message, option RETURNING
    unicode_stego

    OUT unicode_stego
```

Gambar 4.13 Implementasi Pergeseran Rarangkén

Rincian fungsi shiftRarangkén ditunjukkan pada Gambar 4.14. Sistem melakukan perulangan sebanyak panjang cover_stripped. Jika sistem menemukan rarangkén, maka sistem melakukan penyisipan binary_char (*array of char*) yang merupakan bit *secret message* seperti yang ditunjukkan pada baris 10, Jika option yang dipilih adalah 1 (*shift all*), maka sistem memanggil countShiftAll untuk menghitung jumlah rarangkén dari aksara ngalagena yang harus digeser. Hasil tersebut mempengaruhi indeks pembacaan teks. Jika option yang dipilih adalah 0 (*shift group*) maka sistem menghitung secara normal, karena setiap kombinasi rarangkén yang berpasangan diwakili satu nilai Unicode, kecuali untuk kombinasi rarangkén panolong dan pangwisad seperti ditunjukkan pada baris 19. Akumulasi stego_message (string) yang telah digeser kemudian dilakukan proses stripMessage. Hasil stripMessage kemudian menjadi nilai yang dikembalikan.

```
1   FUNCTION          shiftRarangkén(cover_stripped,
    binary_message, option)

2   stego_message ← ""
```

```
3   stego_stripped ← []
4   binary_char ← []

5   binary_message←binary_message.replace(" ", "")
6   binary_char ← binary_message.toCharArray(
7   j ← 0

8
9   FOR i in range of cover_stripped.length
        IF CALL checkRarangkén with
    cover_stripped[i] RETURNING status is true and j
    < binary_char.length THEN
10      stego_message ← stego_message + CALL
    shift with i, 1, binary_char[j], cover_stripped
    RETURNING stego_message
11      IF option is 1 THEN
12          IF cover_stripped[i] is "1ba6" THEN
13              stego_message ← stego_message +
    "\\u" + cover_stripped[i+1]
14              increment i
15          counter ← CALL countShiftAll with i,
    cover_stripped RETURNING counter
16          stego_message ← stego_message + CALL
    shift with i+1, counter, binary_char[j],
    cover_stripped RETURNING stego_message
17          i ← i + counter;
18
19      ELSE
            IF cover_stripped[i] is "1ba7" and
    cover_stripped[i+1] is "1b82" THEN
20              stego_message ← stego_message +
    CALL shift with i+1, counter, binary_char[j],
    cover_stripped RETURNING stego_message
21              increment i
22
23          increment j
24      ELSE
            stego_message ← stego_message + "\\u" +
    cover_stripped[i]
25  stego_stripped ← CALL stripMessage with
    stego_message RETURNING stripped

26
    RETURN stego_stripped

27
    ENDFUNCTION
```

Gambar 4.14 Implementasi Fungsi shiftRarangkén

4.3.3.1 Implementasi Fungsi shift

Fungsi shift bertujuan untuk menentukan apakah suatu rarangkén mengalami pergeseran atau tidak. Sistem melakukan perulangan selama nilai counter (*int*) yang merupakan jumlah rarangkén yang harus digeser. Jika binary (*char*) bernilai 1, maka sistem melakukan pergeseran dengan mengganti rarangkén pada index saat itu. Jika binary bernilai 0, maka sistem tidak melakukan pergeseran dan hanya mengembalikan message normal. *Pseudocode* fungsi ditunjukkan pada Gambar 4.15

```
1   FUNCTION shift(index, counter, binary,
    cover_stripped)
2   stego_message <-- ""
3   FOR i in range of counter
4       IF binary is '1' THEN
5          stego_message <-- stego_message + CALL
    replaceRarangkén with cover_stripped[index]
    RETURNING replaced_rarangkén
7       ELSE
8          stego_message <-- stego_message + "\\u" +
    cover_stripped[index]
9          increment index
10  RETURN stego_message
11  ENDFUNCTION
```

Gambar 4.15 Implementasi Fungsi shift

4.3.3.2 Implementasi Fungsi replaceRarangkén

Funsi replaceRarangkén untuk mengganti nilai Unicode lama dengan nilai Unicode modifikasi yang telah digeser. Sistem menggunakan *switch case* untuk menangani penggantian ini. Pseudocode fungsi ditunjukkan pada Gambar 4.16.

```
1   FUNCTION replaceRarangkén(stripped)
2   replaced_rarangkén <-- "\\u"
3   Case based on stripped
4        case old unicode value
5          replaced_rarangkén += new unicode value
6        default
7   return replaced_rarangkén
8   ENDFUNCTION
```

Gambar 4.16 Implementasi Fungsi replaceRarangkén

4.3.4 Implementasi Penyimpanan *Stego Text*

Proses penyimpanan *stego text* menggunakan data unicode_stego (*array of string*). Nilai unicode_stego kemudian diubah ke dalam teks melalui fungsi getText menjadi stego_text (string). Penyimpanan stego_text ke dalam bentuk PDF dilakukan dengan fungsi generatePDF. Pseudocode proses secara umum ditunjukkan pada Gambar 4.17.

```
1  READ unicode_stego

2  String stego_text = getText(unicode_stego);
3  generatePDF(stego_text, "stego_text.pdf");

4  OUT stego_text.pdf
```

Gambar 4.17 Implementasi Penyimpanan Stego Text

Rincian fungsi getText ditunjukkan pada Gambar 4.18. Sistem melakukan perulangan sebanyak panjang cover_stripped. Setiap perulangan, sistem mengubah nilai cover_stripped (*string*) menjadi nilai hex dalam hex_val (*int*). Nilai hex_val kemudian diubah ke dalam bentuk teks dan disimpan dalam variable text (*string*). Fungsi mengembalikan nilai text.

```
1  FUNCTION getText (cover_stripped)

2  text <-- ""
3  hex_val <-- 0

4  FOR i in range of cover_stripped length
5      hex_val <-- parseInt cover_stripped[i] to HEX
6      text <-- text + cast hex_val to char

7  return text

8  ENDFUNCTION
```

Gambar 4.18 Implementasi Fungsi getText

Rincian fungsi generatePDF ditunjukkan pada Gambar 4.19. Sistem mengawali fungsi generatePDF dengan pembuatan PDDocument baru sebagai doc. Selanjutnya, sistem membuat kelas

PDFRenderingSystem baru dengan nama renderer. Sistem membagi message berdasarkan baris menjadi message_line (*array of string*). Fungsi renderText dalam kelas PDFRenderingSystem dipanggil untuk memulai penulisan *stego text* dalam doc. Setelah selesai renderText, sistem menutup renderer dan menyimpan doc ke dalam path yang sudah ditentukan.

```
1   FUNCTION generatePDF(message, path)

2   doc <-- new PDDocument
3   renderer <-- new PDFRenderingSystem from doc
4   message_line <-- []
5   message_line <-- message.split("\n")

6   CALL renderText with message_line, 50 from
7   renderer

8   close renderer
9   save doc to path
10  close doc
11  ENDFUNCTION
```

Gambar 4.19 Implementasi Fungsi generatePDF

4.3.4.1 Implementasi Kelas PDFRenderingSystem

Kelas PDFRenderingSystem bertujuan untuk menangani proses penulisan dan penyimpanan teks ke dalam bentuk PDF. Kelas ini memiliki 3 variabel global yaitu doc, content, dan textRenderingLineY. Selain itu, terdapat 4 fungsi yaitu PDFRenderingSystem, newPage, renderText, dan close. *Pseudocode* kelas ditunjukkan pada Gambar 4.20

```
1   CLASS PDFRenderingSystem:
2   doc <-- null
3   content <-- null
4   textRenderingLineY <-- 0
5   FUNCTION PDFRenderingSystem(doc)
6   FUNCTION newPage()
7   FUNCTION           renderText(message_stripped,
    margin_width)
7   FUNCTION close()
8   END CLASS
```

Gambar 4.20 Implementasi Kelas PDFRenderingSystem

4.3.4.2 Implementasi Fungsi PDFRenderingSystem

Fungsi PDFRenderingSystem merupakan konstruktor dari kelas PDFRenderingSystem. Fungsi ini bertujuan untuk membuat dokumen baru. Sistem melakukan pengisian variabel global doc dengan doc yang menjadi argumen masukan. *Pseudocode* fungsi ditunjukkan pada Gambar 4.21,

```
1  FUNCTION PDFRenderingSystem(doc)
2  READ this.doc

3  this.doc <-- doc

4  ENDFUNCTION
```

Gambar 4.21 Implementasi Fungsi PDFRenderingSystem

4.3.4.3 Implementasi Fungsi newPage

Fungsi newPage bertujuan untuk menambah halaman baru. Dalam fungsi ini, sistem membuat PDPage berukuran A4 dan memasukannya ke dalam doc. Selanjutnya, sistem membuat PDPageContentStream dalam doc. Selain itu, dilakukan juga inisialisasi posisi textRenderingLineY (*int*) yang merupakan posisi penulisan teks menjadi di atas. *Pseudocode* fungsi ditunjukkan pada Gambar 4.22.

```
1  FUNCTION newPage()
2  READ doc, content, textRenderingLineY

3  CALL close

4  page <-- new PDPage size A4
5  add this page to doc
6  content <-- new PDPageContentStream in doc, page
7  textRenderingLineY <-- 780
8  ENDFUNCTION
```

Gambar 4.22 Implementasi Fungsi newPage

4.3.4.4 Implementasi Fungsi renderText

Fungsi renderText bertujuan untuk melakukan penulisan teks ke dalam dokumen. Sistem mengawali fungsi ini dengan memanggil newPage untuk membuat halaman baru. Selanjutnya,

sistem membuka berkas *font* yang digunakan dan menyimpannya ke dalam variabel *font*. Sistem melakukan pengaturan ukuran *font* dan posisi penulisan awal. Setelah lingkungan selesai, sistem melakukan perulangan sebanyak panjang message_stripped (*array of string*). Setiap perulangan sistem memeriksa variabel textRenderingLineY (*int*) yang merupakan penanda posisi. Jika textRenderingLineY lebih dari sama dengan 60, maka sistem melakukan penulisan message_stripped pada baris tersebut. Selanjutnya, sistem mengurangi nilai textRenderingLineY dan berpindah baris posisi. Jika textRenderingLineY kurang dari 60, maka sistem menambah halaman baru dan mengatur ulang *font* dan posisi textRenderingLineY. *Pseudocode* fungsi ditunjukkan pada Gambar 4.23.

```
1   FUNCTION renderText(message_stripped,
    margin_width)
2   READ doc, content, textRenderingLineY

3   CALL newPage

4   font<-- load font SundaneseUnicode2013Custom.ttf
5   beginText in content
6   setFont with font, size=18 in content
7   newLineAtOffset with margin_width,
    textRenderingLineY in content

8   FOR i in range of message_stripped.length
9     IF textRenderingLineY < 60 THEN
10         CALL newPage
11         beginText in content
12         setFont with font, size=18 in content
13         newLineAtOffset with margin_width,
    textRenderingLineY in content

14    textRenderingLineY <-- textRenderingLineY - 30
15    showText with message_stripped[i] in content
16    newLineAtOffset with 0, -30 in content

17  endText in content

18  ENDFUNCTION
```

Gambar 4.23 Implementasi Fungsi renderText

4.3.4.5 Implementasi Fungsi close

Fungsi close bertujuan untuk mengosongkan nilai content untuk halaman baru. Jika content untuk halaman baru tidak kosong, sistem menutup content tersebut, lalu mengisinya dengan nilai kosong. Fungsi ini digunakan untuk pergantian halaman. *Pseudocode* fungsi ditunjukkan pada Gambar 4.24.

```
1  FUNCTION close()
2  READ content

3  IF content is not null
4       content.close()
5       content <-- null
6  ENDFUNCTION
```

Gambar 4.24 Implementasi Fungsi close

4.4 Implementasi Modul Extracting

Implementasi modul *extracting* secara umum terbagi menjadi 3 proses utama yaitu pembacaan *stego tex*t, penghitungan panjang *secret message*, dan ekstraksi *secret message*. Keterangan implementasi tiap proses, dijelaskan pada masing-masing sub-bab.

4.4.1 Implementasi Pembacaan Stego Text

Proses pembacaan stego text menerima data masukan stego_text dalam bentuk PDF. Masukan ini dibaca ketika fungsi openStegoText dipanggil. Secara umum, proses pembacaan *stego text* dilakukan dengan memanggil fungsi readStegoText. Hasil pembacaan disimpan dalam variabel stego_text (*string*). *Pseudocode* proses secara umum ditunjukkan pada Gambar 4.25.

```
1  stego_text <-- ""
2  stego_text  <--  CALL  readStegoText  RETURNING
3  stego_text
```

Gambar 4.25 Implementasi Pembacaan Stego Text

Rincian fungsi readStegoText ditunjukkan pada Gambar 4.26. Sistem memanggil fungsi openStegoText untuk mendapatkan

stego_text. Selanjutnya, stego_text diubah ke dalam nilai Unicode melalui fungsi getUnicodeValue menjadi unicode_stego (*array of string*).

```
1  FUNCTION readStegoText()

2  stego_text <-- ""
3  unicode_stego <-- ""

4  stego_text <-- CALL openStegoText RETURNING
   stego_text
5  unicode_stego <-- CALL getUnicodeValue with
   stego_text RETURNING unicode_value

6  return unicode_stego
7  ENDFUNCTION
```

Gambar 4.26 Implementasi Fungsi readStegoText

4.4.1.1 Implementasi Fungsi openStegoText

Fungsi openStegoText bertujuan untuk membuka dan membaca berkas *stego text*. Sistem membuka PDDocument dari berkas yang berasal dari *location (string)* tertentu. Masukan *location* didapat dari antarmuka sistem. Hasil tersebut disimpan dalam variabel *document (PDDocument)*. Selanjutnya, sistem membuat PDFTextStripper dan mengambil teks menggunakan PDFTextStripper. Hasil tersebut disimpan dalam variabel *text (string)*. Fungsi mengembalikan *text*. *Pseudocode* fungsi ditunjukkan pada Gambar 4.27.

```
1  FUNCTION openStegoText()
2  READ location

3  path <-- location
4  file <-- new File from path
5  text <-- ""
6  document <-- load PDDocument from file
7  pdfstripper <-- new PDFTextStripper
8  text <-- pdfStripper.getText from document

9  return text
10 ENDFUNCTION
```

Gambar 4.27 Implementasi Fungsi openStegoText

4.4.1.2 Implementasi Fungsi getUnicodeValue

Fungsi getUnicodeValue bertujuan untuk mengubah nilai *stego text* menjadi nilai Unicode. Sistem mengawali fungsi dengan mengubah stego_text (*string*) ke dalam bentuk *array of char* menjadi unicode_char (*array of char*). Selanjutnya, sistem melakukan perulangan untuk char dalam unicode_char. Setiap perulangan, sistem mengubah setiap nilai *char* menjadi nilai heksadesimal. Nilai heksadesimal ditambahkan "\\u" di awal untuk mendapatkan nilai Unicode. Akumulasi nilai Unicode disimpan pada variabel unicode_stego (*string*). Fungsi mengembalikan unicode_stripped (*array of string*) yaitu nilai Unicode yang telah melalui fungsi stripMessage. *Pseudocode* fungsi ditunjukkan pada Gambar 4.28.

```
1   FUNCTION getUnicodeValue(stego_text)
2   unicode_stego <-- ""
3   unicode_char[] <-- toCharArray(stego_text)
4   unicode_stripped <-- []

5   FOR c in unicode_char
6         unicode_stego <-- unicode_stego + "\\u" +
    toHexString(c)

7   unicode_stripped  <--  CALL  stripMessage  with
8   unicode_stego RETURNING stripped

9   RETURN unicode_stripped
10  ENDFUNCTION
```

Gambar 4.28 Implementasi Fungsi getUnicodeValue

4.4.2 Implementasi Penghitungan Panjang Secret Message

Proses penghitungan panjang *secret message* menggunakan data stego_text (*string*) dan option (*int*). Proses penghitungan ini bersamaan dengan proses ekstraksi *secret message*. Data stego_text dilakukan stripMessage untuk mendapatkan stego_stripped (*array of string*) atau nilai unicode per huruf. Selanjutnya, sistem memanggil fungsi readRarangkén yang mengembalikan isi secret_message (*string*) sekaligus menghitung

panjangnya. Threshold (*int*) merupakan batas bit panjang *secret message* yang digunakan. Contohnya threshold yang digunakan adalah 8 bit. *Pseudocode* proses secara umum ditunjukkan pada Gambar 4.29.

```
1  READ stego_text option

2  secret_message <-- “”
3  stego_stripped <-- []

4  stego_stripped <-- CALL stripMessage with
   stego_text RETURNING stripped

5  secret_message <-- CALL readRarangkén with
   stego_stripped, index=1, threshold=8, option
   RETURNING secret_message

6  OUT secret_message
```

Gambar 4.29 Implementasi Penghitungan Panjang Secret Message

Rincian fungsi readRarangkén ditunjukkan pada Gambar 4.30, Sistem mengawali fungsi dengan inisialisasi nilai i (*int*) sesuai dengan masukan index (*int*). Sistem melakukan perulangan selama terdapat threshold. Ketika sistem menemukan rarangkén, maka sistem menggunakan fungsi getSecretBit untuk mendapatkan *secret bit* lalu memasukannya ke dalam secret_message (*string*). Jika sistem menggunakan option 1 (*shift all*), sistem melewati sisa rarangkén dalam aksara ngalagena saat itu. Jika sistem menggunakan option 0 (*shift group*), sistem melewati rarangkén sesuai dengan kelompok rarangkén. Setelah threshold yang terus berkurang mencapai 0, maka sistem berhenti melakukan perulangan. Selanjutnya, sistem mengubah bit yang didapat menjadi integer panjang *secret message* yang digunakan untuk proses ekstraksi *secret message*. Proses ekstraksi *secret message* yang dilakukan menggunakan panjang *secret message* sebagai threshold dan melanjutkan index dari i pada saat itu. Proses ini dipanggil dengan rekursif pada readRarangkén.

```
1   FUNCTION readRarangkén(stego_stripped, index,
    threshold, option)

2   i <-- index
3   secret_message <-- ""
4   WHILE threshold > 0
5   DO
6      IF CALL checkAllRarangkén with stego_stripped
    RETURNING status is true THEN
7         secret_message <-- secret_message + CALL
    getSecretBit with stego_stripped RETURNING
    secret_bit

7          IF stego_stripped[i] is "1ba6" or
    stego_stripped[i] is "1bd2" THEN
8             increment i

9          IF option is 1 THEN
10            i <-- i + CALL countShiftAll with
    i stego_stripped RETURNING counter
11         ELSE
12            IF (stego_stripped[i] is "1ba7"
    and stego_stripped[i+1] is "1b82") or
    (stego_stripped[i] is "1bd3" and
    stego_stripped[i+1] is "1bcc") THEN
13            increment i

14          decrement threshold
15     increment i

16  IF index is 1 THEN
17     threshold <-- parseInt secret_message from
    BINARY
18     secret_message <-- CALL readRarangkén with
    stego_stripped, i, threshold, option

19  RETURN secret_message

20  ENDFUNCTION
```

Gambar 4.30 Implementasi Fungsi readRarangkén

4.4.2.1 Implementasi Fungsi checkAllRarangkén

Fungsi checkAllRarangkén bertujuan untuk memeriksa nilai Unicode. Jika nilai unicode yang diperiksa terdapat rarangkén geser melalui fungsi checkShiftedRarangkén atau rarangkén normal melalui checkRarangkén, maka sistem mengembalikan nilai *true*. Jika tidak, maka sistem mengembalikan nilai *false*. *Pseudocode* fungsi ditunjukkan pada Gambar 4.31

```
1  FUNCTION checkAllRarangkén(stego_stripped)

2  IF CALL checkShiftedRarangkén with
   stego_stripped RETURNING status is true THEN
3     RETURN true
4  ELSE IF CALL checkRarangkén with stego_stripped
   RETURNING status is true THEN
5     RETURN true
6  ELSE
7     RETURN false

8  ENDFUNCTION
```

Gambar 4.31 Implementasi Fungsi checkAllRarangkén

4.4.2.2 Implementasi Fungsi checkShiftedRarangkén

Fungsi checkShiftedRarangkén bertujuan untuk mengetahui rarangkén yang digeser. Sistem mengetahui hal tersebut dengan mengubah nilai Unicode menjadi nilai integer heksadesimal. Jika nilai integer heksadesimal berada pada rentang 7114-7136, maka sistem mengembalikan nilai *true*. Jika tidak, maka sistem mengembalikan nilai *false*. *Pseudocode* fungsi ditunjukkan pada Gambar 4.32.

```
1  FUNCTION checkShiftedRarangkén(stripped)

2  hex_val ← 0
3  hex_val ← parseInt(stripped) to HEX
4  return hex_val >= 7114 and hex_val <= 7136

5  ENDFUNCTION
```

Gambar 4.32 Implementasi Fungsi CheckShiftedRarangkén

4.4.2.3 Implementasi Fungsi checkRarangkén

Fungsi checkRarangkén bertujuan untuk mengetahui rarangkén normal. Sistem mengetahui hal tersebut dengan mengubah nilai unicode menjadi nilai integer heksadesimal. Jika nilai integer heksadesimal berada pada rentang 7040-7042, 7073-7082 atau 7104-7112, maka sistem mengembalikan nilai *true*. Jika tidak, maka sistem mengembalikan nilai *false*. Pseudocode fungsi ditunjukkan pada Gambar 4.33.

```
1  FUNCTION checkRarangkén(stripped)
2  hex_val <-- 0
3  hex_val <-- parseInt(stripped) to HEX
4  return (hex_val >= 7040 and hex_val <= 7042) or
   (hex_val >= 7073 and hex_val <= 7082) or (hex_val
   >=7104 and hex_val <=7112)

5  ENDFUNCTION
```

Gambar 4.33 Implementasi Fungsi checkRarangkén

4.4.2.4 Implementasi Fungsi getSecretBit

Fungsi getSecretBit bertujuan untuk mendapatkan secret bit yang merupakan bagian dari secret message. Sistem mendapatkan secret bit dengan cara memeriksa apakah rarangkén mengalami pergeseran atau tidak. Jika rarangkén mengalami pergeseran, maka secret bit yang didapatkan adalah 1, Jika tidak, maka secret bit yang didapatkan adalah 0, Pseudocode fungsi ditunjukkan pada Gambar 4.34

```
1  FUNCTION getSecretBit(stego_stripped)
2  secret_bit <-- ""

3  IF     CALL     checkShiftedRarangkén     with
   stego_stripped RETURNING status is true THEN
4      secret_bit <-- "1"
5  ELSE IF CALL checkRarangkén with stego_stripped
   RETURNING status is true THEN
6      secret_bit <-- "0"

7  RETURN secret_bit
8  ENDFUNCTION
```

Gambar 4.34 Implementasi Fungsi getSecretBit

4.4.3 Implementasi Ekstraksi Secret Message

Proses ekstraksi *secret message* menggunakan data stego_text dan option. Proses ini dilakukan ketika proses penghitungan panjang *secret message* pada fungsi readRarangkén (Gambar 4.30) selesai untuk pertama kali. Ketika sistem memanggil fungsi readRarangkén secara rekursif, maka proses ekstraksi *secret message* dimulai. Setelah mendapatkan secret_message (*string*) sistem membagi *secret message* per *byte* (8 bit). Selanjutnya, sistem mengembalikan nilai *byte* menjadi teks *secret message* melalui fungsi getText. Sistem menampilkan keluaran melalui antarmuka modul *extracting. Pseudocode* proses secara umum ditunjukkan pada Gambar 4.35.

```
1  READ stego_stripped option

2  secret_message <-- CALL readRarangkén with
   stego_stripped, index=1, threshold=8, option
   RETURNING secret_message

3  secret_splitted <-- split secret_message into 8
   digit

4  secret_message ←- CALL getText with
   secret_splitted RETURNING secret_message

5  OUT secret_message
```

Gambar 4.35 Implementasi Ekstraksi Secret Message

Rincian fungsi getText ditunjukkan pada Gambar 4.36. Pada fungsi ini, sistem melakukan perulangan untuk mendapatkan secret_message (*string*). Setiap perulangan, sistem mengubah secret_splitted (array of string) yang merupakan *byte* pesan menjadi nilai ASCII. Nilai ASCII tersebut diubah ke dalam bentuk *char* dan disimpan dalam variabel secret_message. Sistem mengembalikan nilai secret_message akhir.

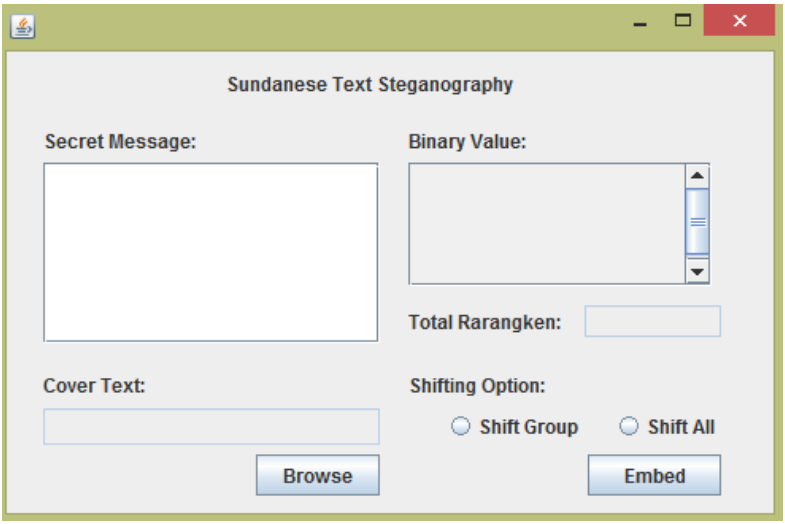
```
1  FUNCTION getText(secret_splitted)

2  secret_message <-- ""
3  ascii_value <-- 0

4  FOR i in range of secret_splitted.length
5        ascii_value          <--          parseInt
   secret_splitted[i] to decimal
6        secret_message <-- secret_message + cast
   ascii_value to char

7  RETURN secret_message

8  ENDFUNCTION
```

Gambar 4.36 Implementasi Fungsi getText

4.5 Implementasi Antarmuka Modul Embedding

Implementasi antarmuka pada modul *embedding* dibangun menggunakan JForm pada lingkungan bahasa pemrograman Java. Tampilan antarmuka mengikuti desain yang telah dirancang pada bab analisis dan perancangan sistem. Antarmuka hasil implementasi modul *embedding* ditunjukkan pada Gambar 4.37. Komponen JForm yang digunakan pada implementasi program terdiri dari:

1. JTextLabel
   Komponen ini digunakan pada setiap judul bagian dari antarmuka.
2. JTextArea
   Komponen ini digunakan untuk tampilan hasil *binary value* dan pengisian *secret message*.
3. JTextField
   Komponen ini digunakan untuk tampilan hasil total rarangkén dan lokasi *cover text*.
4. Button
   Komponen ini digunakan untuk tombol *browse* lokasi *cover text* dan *embed cover text.*
5. Radio Button
   Komponen ini digunakan untuk pemilihan *shifting option.*



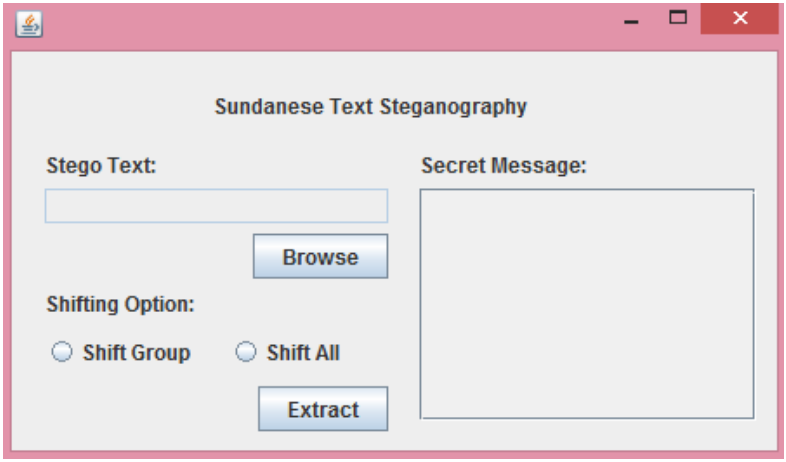
Gambar 4.37 Implementasi Antarmuka Modul Embedding

4.6 Implementasi Antarmuka Modul Extracting

Implementasi antarmuka pada modul extracting dibangun menggunakan JForm pada lingkungan bahasa pemrograman Java. Tampilan antarmuka mengikuti desain yang telah dirancang pada bab analisis dan perancangan. Antarmuka hasil implementasi modul *extracting* ditunjukkan pada Gambar 4.38. Komponen JForm yang digunakan pada implementasi program terdiri dari:

1. JTextLabel
   Komponen ini digunakan pada setiap judul bagian dari antarmuka
2. JTextArea
   Komponen ini digunakan untuk tampilan hasil pencarian lokasi *stego text.*
3. JTextField
   Komponen ini digunakan untuk tampilan hasil ekstraksi *secret message*

4. Button
   Komponen ini digunakan untuk tombol browse *lokasi stego text* dan *embed cover text.*
5. Radio Button
   Komponen ini digunakan untuk pemilihan *shifting option.*



Gambar 4.38 Implementasi Antarmuka Modul Extracting

BAB V
UJI COBA DAN EVALUASI

Bab ini menguraikan tentang uji coba dan evaluasi sistem steganografi teks pada aksara Sunda. Uji coba dilakukan pada modul *embedding* dan *extracing* berdasarkan skenario yang ada. Uji coba meliputi uji coba fungsionalitas, panjang masukan, perubahan ukuran, waktu dan *capacity ratio.*

5.1 Lingkungan Uji Coba

Uji coba sistem steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding* menggunakan perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 5.1,

Tabel 5.1 Lingkungan Uji Coba Perangkat Lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i5-4200U CPU @ 1,60GHz
	Memori	4GB 1600 MHz DDR3
Perangkat Lunak	Sistem Operasi	Windows 8.1
	Bahasa Pemrograman	Java
	Perangkat Pengembang	NetBeans IDE 8.1
	Word Editor	Ms. Word 2013

5.2 Data Uji Coba

Data uji coba sistem steganografi teks pada aksara Sunda adalah teks aksara sunda dalam bentuk docx sebagai *cover text*. Ukuran *cover text* dalam bentuk PDF digunakan untuk uji coba perubahan ukuran dan *capacity ratio*. Teks aksara Sunda yang digunakan merupakan teks puisi. Rincian setiap data uji coba ditunjukkan pada Tabel 5.2.

Tabel 5.2 Data Uji Coba

No	Nama Berkas	Ukuran docx (kB)	Ukuran pdf (kB)	Sumber
1	Halodo Teuing ku Panjang.docx	4,79	7,17	Internet [24]
2	Itungan Samemeh Balitungan.docx	5,09	8,24	Internet [25]
3	Sekar Merdika.docx	4,88	6,92	Internet [26]
4	Walungan.docx	5,37	8,48	Internet [27]
5	Baris Kahiji.docx	5,25	8,59	Internet [28]
6	Pupujian Nabi Urang .docx	8,93	14,78	Internet [29]
7	Kumpulan Puisi Godi Suwarna.docx	18,3	19,20	Internet [30]
8	Jante Arkidam.docx	5,53	8,96	Internet [31]
9	Sajak-sajak Lia Taliah.docx	5,57	9,88	Internet [32]
10	Antologi Pasanggiri Maca Sajak Sunda 2015.docx	9,50	16,62	Internet [33]

5.3 Skenario Uji Coba

Uji coba dilakukan untuk mengetahui ketercapaian dan performa sistem steganografi teks pada aksara Sunda. Ketercapaian dapat dilihat dari beberapa aspek yaitu fungsionalitas, panjang *secret message*, perubahan ukuran, waktu dan *capacity ratio*. Setiap uji coba memiliki skenario untuk mencoba kemungkinan dan mengetahui nilai yang sesuai untuk diterapkan.

5.3.1 Skenario Uji Coba Fungsionalitas

Uji coba fungsionalitas bertujuan untuk mengetahui fungsionalitas yang dapat berjalan pada sistem. Fungsionalitas mencakup proses *embedding* dan *extracting* pada *shifting option* yaitu *shift all* dan *shift group*. Skenario untuk uji coba fungsionalitas terdiri dari:

A. Proses *embedding* dan *extracting* dengan masukan secret message "saya rijal" menggunakan pergeseran *shift all* pada data ke-1,
B. Proses *embedding* dan *extracting* dengan masukan *secret message* "saya rijal" menggunakan pergeseran *shift group* pada data ke-1,
C. Proses embedding dan extracting dengan masukan *secret message* "saya berumur 17 tahun!" menggunakan pergeseran *shift all* pada data ke-4.
D. Proses *embedding* dan *extracting* dengan masukan *secret message* "saya berumur 17 tahun!" menggunakan pergeseran *shift group* pada data ke-4.

5.3.2 Skenario Uji Coba Panjang Secret Message

Uji coba panjang *secret message* bertujuan untuk mengetahui batas karakter yang dapat menjadi masukan. Batas karakter dipengaruhi oleh bit penanda panjang *secret message* yang ditambahkan di awal bit isi *secret message*. Parameter normal

8 bit penanda panjang *secret message* diubah pada skenario ini. Hasil yang akan ditinjau adalah minimal rarangkén yang dibutuhkan untuk satu karakter dan maksimal jumlah karakter yang dapat menjadi masukan. Skenario terdiri dari:

A. Penanda panjang *secret message* = 8 bit
B. Penanda panjang *secret message* = 9 bit
C. Penanda panjang *secret message* = 10 bit
D. Penanda panjang *secret message* = 11 bit
E. Penanda panjang *secret message* = 12 bit

5.3.3 Skenario Uji Coba Perubahan Ukuran

Uji coba perubahan ukuran bertujuan untuk mengetahui perubahan ukuran dari *cover text* menjadi *stego text*. Untuk pengujian ini, PDF yang dihasilkan oleh sistem tidak hanya *stego text*, tetapi juga *cover text.* Parameter yang diubah pada uji coba ini meliputi secret message, *shifting option* dan algoritma pergeseran rarangkén secara acak. Setiap berkas, skenario terdiri dari:

A. Proses *embedding* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift all* dan sisa rarangkén digeser secara acak.
B. Proses *embedding* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift group* dan sisa rarangkén digeser secara acak.
C. Proses *embedding* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift all* dan sisa rarangkén digeser secara acak.
D. Proses *embedding* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift group* dan sisa rarangkén digeser secara acak.
E. Proses *embedding* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift all* dan sisa rarangkén tidak digeser secara acak.
F. Proses *embedding* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift group* dan sisa rarangkén tidak digeser secara acak.

G. Proses *embedding* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift all* dan sisa rarangkén tidak digeser secara acak.
H. Proses *embedding* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift group* dan sisa rarangkén tidak digeser secara acak.

5.3.4 Skenario Uji Coba Waktu

Uji coba waktu bertujuan untuk mengetahui waktu yang diperlukan untuk melakukan proses *embedding* dan *extracting.* Waktu yang dihitung hanya waktu pada proses utama *embedding* yaitu penyisipan dengan pergeseran rarangkén dan *extracting* yaitu ekstraksi informasi. Parameter yang diubah pada uji coba ini meliputi *secret message*, *shifting option* dan algoritma pergeseran rarangkén secara acak. Setiap data, skenario terdiri dari:

A. Waktu *embedding* dan *extracting* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift all* dan sisa rarangkén digeser secara acak.
B. Waktu *embedding* dan *extracting* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift group* dan sisa rarangkén digeser secara acak.
C. Waktu *embedding* dan *extracting* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift all* dan sisa rarangkén digeser secara acak.
D. Waktu *embedding* dan *extracting* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift group* dan sisa rarangkén digeser secara acak.
E. Waktu *embedding* dan *extracting* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift all* dan sisa rarangkén tidak digeser secara acak.
F. Waktu *embedding* dan *extracting* dengan masukan *secret message* "rijal" menggunakan pergeseran *shift group* dan sisa rarangkén tidak digeser secara acak.

G. Waktu *embedding* dan *extracting* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift all* dan sisa rarangkén tidak digeser secara acak.
H. Waktu *embedding* dan *extracting* dengan masukan *secret message* "nama saya rijal" menggunakan pergeseran *shift group* dan sisa rarangkén tidak digeser secara acak.

5.3.5 Skenario Uji Coba Capacity Ratio

Uji *capacity ratio* bertujuan untuk mengetahui kemampuan cover text dalam menyimpan bit *secret message*. *Capacity ratio* dihitung dengan membagi total rarangkén (bit) dan ukuran *cover text* (kiloByte). Pada pengujian ini besaran hasil capacity ratio adalah bit/kiloByte. Parameter yang digunakan adalah *shifting option*. Setiap berkas, skenario terdiri dari:

A. *Capacity ratio* dengan masukan *shift all* pada *shifting option*.
B. *Capacity ratio* dengan masukan *shift group* pada *shifting option*.

5.4 Hasil Uji Coba

Hasil uji coba berisi tentang hasil uji coba dari skenario yang telah dilakukan. Hasil-hasil skenario uji coba dibandingkan untuk dilakukan analisis dan evaluasi.

5.4.1 Hasil Uji Coba Fungsionalitas

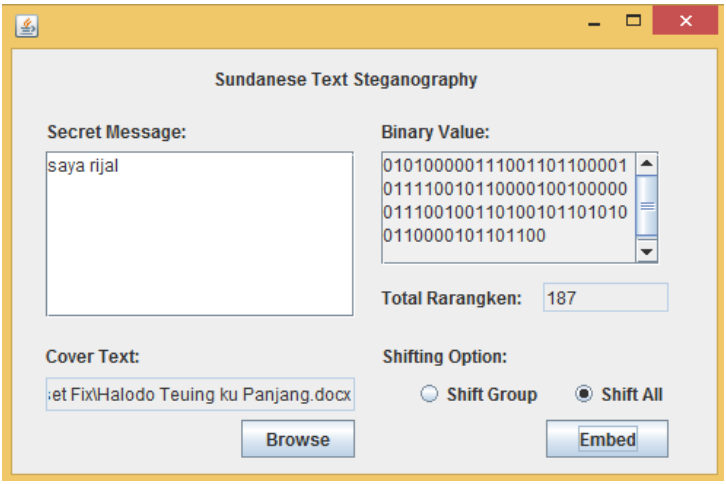
Hasil uji coba fungsionalitas berisi tentang eksekusi jalannya program. Fungsionalitas yang diuji ditunjukkan pada Tabel 5.3. Secara umum, fungsionalitas yang diuji pada *modul embedding* dan *extracting* dapat berjalan dengan baik. Rincian eksekusi fungsionalitas setiap skenario dijelaskan pada masing-masing sub-bab.

Tabel 5.3 Hasil Uji Coba Fungsionalitas

No.	Fungsionalitas	Keterangan
1,	Konversi secret message	Sukses
2.	Penghitungan total rarangkén	Sukses
3.	Embedding secret message	Sukses
4.	Penyimpanan stego text	Sukses
5.	Ekstraksi secret message	Sukses

5.4.1.1 Hasil Uji Coba Fungsionalitas Skenario A

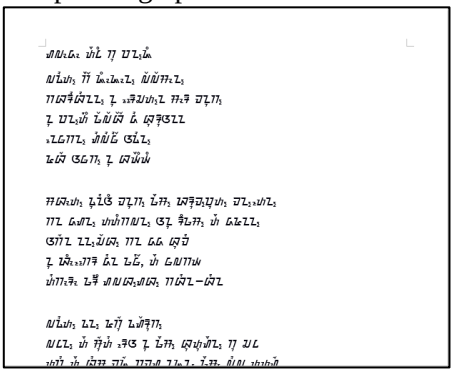
Hasil uji coba fungsionalitas skenario A pada proses *embedding* ditunjukkan pada Gambar 5.1, Setelah mengisi data masukan sesuai skenario, sistem memunculkan hasil *binary value* dari *secret message*. Selain itu, terdapat total rarangkén sesuai *shifting option* yang dipilih. Pada skenario A, data ke-1 memiliki 187 rarangkén jika dihitung menggunakan pilihan *shift all*.



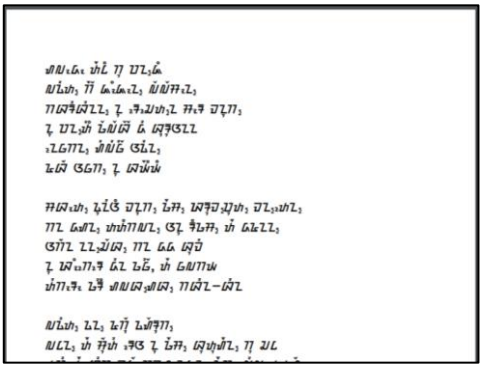
Gambar 5.1 Antarmuka Modul *Embedding* Skenario A

Keluaran proses *embedding* adalah stego.pdf. Tampilan stego.pdf ditunjukkan pada Gambar 5.3. Hasil teks tidak jauh

berbeda dengan masukan Halodo Teuing ku Panjang.docx yang ditunjukkan pada Gambar 5.2. Kedua berkas, Halodo Teuing ku Panjang.docx maupun stego.pdf memiliki satu halaman.



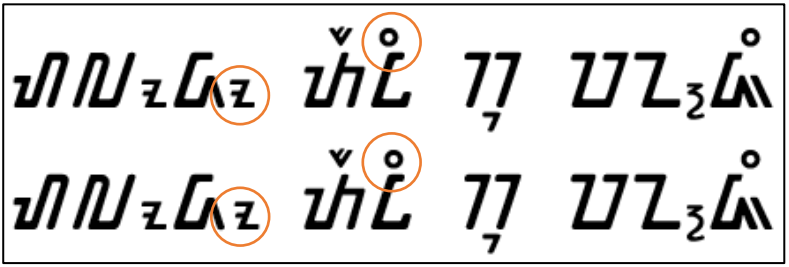
Gambar 5.2 Masukan docx Skenario A



Gambar 5.3 Keluaran stego.pdf Skenario A

Perbandingan ditunjukkan pada gambar 5.4. Perbandingan tersebut diambil dari baris pertama *cover text* dan *stego text.* Karena proses *embedding*, teks dari stego.pdf mengalami pergeseran pada beberapa rarangkén. Rarangkén yang mengalami pergeseran adalah rarangkén ke-2 dan ke-4. Pergeseran tersebut sesuai dengan 4 bit pertama dari bit *secret message* yaitu 0101, Bit

0 menunjukan tidak ada pergeseran yang akan dilakukan, sedangkan bit 1 menunjukan ada pergeseran yang akan dilakukan.



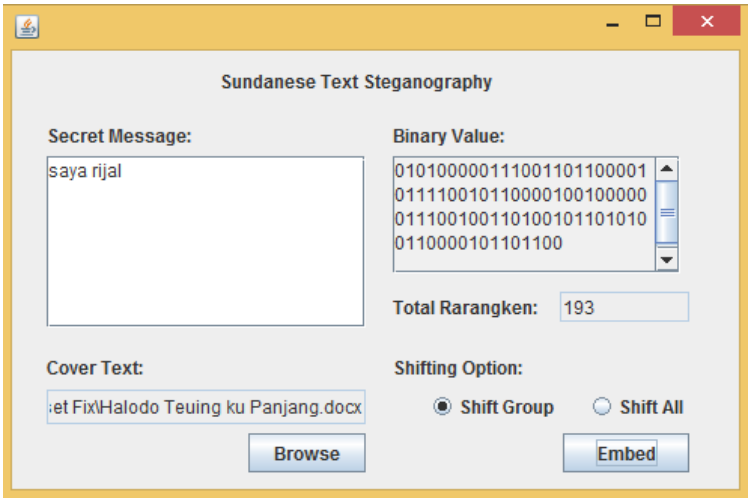
Gambar 5.4 Perbandingan Cover (Atas) dan Stego (Bawah) Skenario A

Hasil modul extracting pada Skenario A ditunjukkan pada Gambar 5.5. Dengan menggunakan *shifting option* yang sama dengan modul *embedding* yaitu *shift all*, hasil keluaran *secret message* adalah "saya rijal". *Secret message* ini sama dengan *secret message* yang dimasukan pada modul *embedding.*

Gambar 5.5 Antarmuka Modul Extracting Skenario A

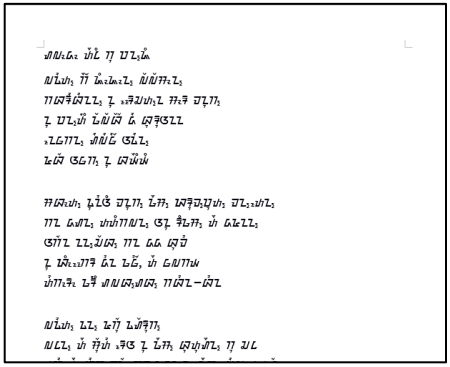
5.4.1.2 Hasil Uji Coba Fungsionalitas Skenario B

Hasil uji coba fungsionalitas skenario B pada proses embedding ditunjukkan pada Gambar 5.6. Setelah mengisi data masukan sesuai skenario, sistem memunculkan hasil *binary value* dari *secret message*. Selain itu, terdapat total rarangkén sesuai *shifting option* yang dipilih. Pada skenario B, data ke-1 memiliki 193 rarangkén jika dihitung menggunakan pilihan *shift group.* Hasil ini lebih tinggi dibandingkan hasil pada skenario A.

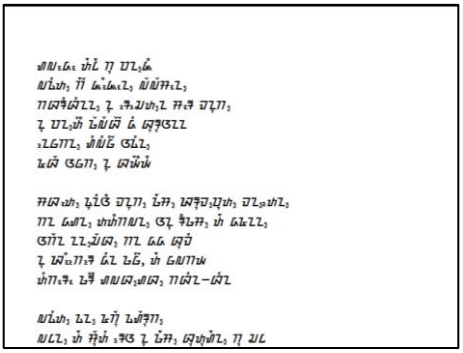


Gambar 5.6 Antarmuka Modul Embedding Skenario B

Keluaran proses *embedding* adalah stego.pdf. Tampilan stego.pdf ditunjukkan pada Gambar 5.3. Seperti skenario A, hasil teks tidak jauh berbeda dengan masukan Halodo Teuing ku Panjang.docx yang ditunjukkan pada Gambar 5.7. Kedua berkas, Halodo Teuing ku Panjang.docx maupun stego.pdf memiliki satu halaman.

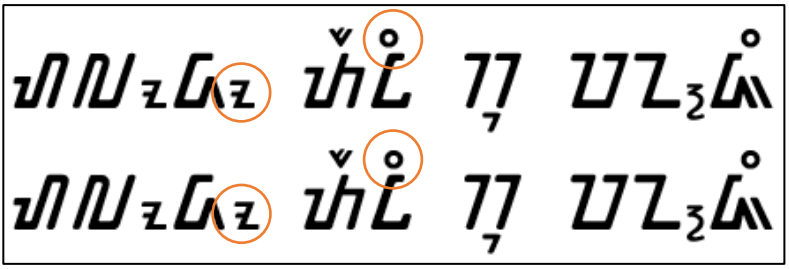


Gambar 5.7 Masukan docx Skenario B



Gambar 5.8 Keluaran stego.pdf Skenario B

Perbandingan ditunjukkan pada gambar 5.9. Perbandingan tersebut diambil dari baris pertama *cover text* dan *stego text*. Karena proses *embedding*, teks stego.pdf mengalami pergeseran pada beberapa rarangkén. Rarangkén yang mengalami pergeseran adalah rarangkén ke-2 dan ke-4. Pergeseran yang dilakukan sama dengan skenario A untuk baris pertama karena aksara ngalagena tidak menpunyai rarangkén yang berpasangan.



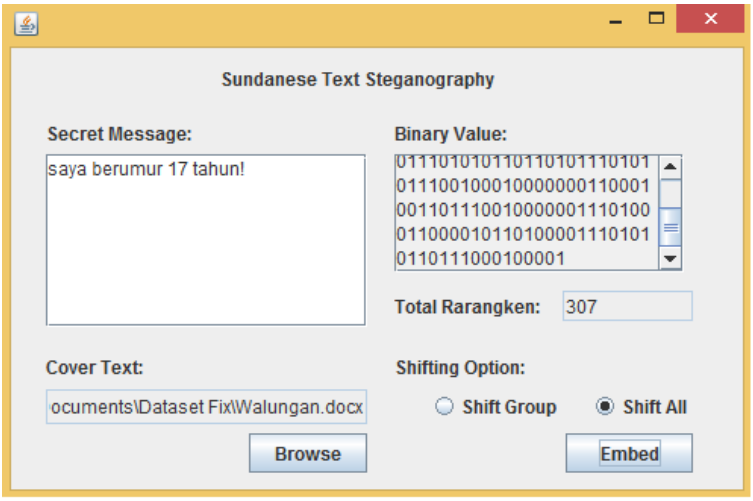
Gambar 5.9 Perbandingan Cover (Atas) dan Stego (Bawah) Skenario B

Hasil modul extracting pada Skenario B ditunjukkan pada Gambar 5.10, Dengan menggunakan *shifting option* yang sama dengan modul *embedding* yaitu *shift group*, hasil keluaran *secret message* adalah "saya rijal". Secret message ini sama dengan *secret message* yang dimasukan pada modul *embedding*.

Gambar 5.10 Antarmuka Modul Extracting Skenario B

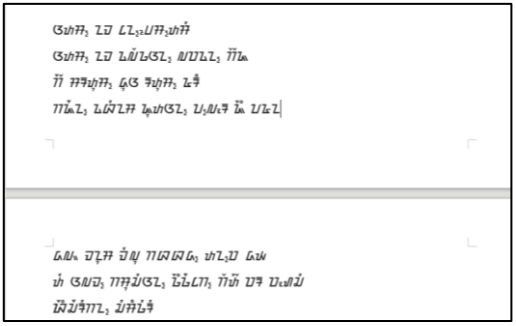
5.4.1.3 Hasil Uji Coba Fungsionalitas Skenario C

Hasil uji coba fungsionalitas skenario C pada proses embedding ditunjukkan pada Gambar 5.11, Setelah mengisi data masukan sesuai skenario, sistem memunculkan hasil *binary value* dari *secret message*. Selain itu, terdapat total rarangkén sesuai *shifting option* yang dipilih. Pada skenario C, data ke-4 memiliki 307 rarangkén jika dihitung menggunakan pilihan *shift all*.

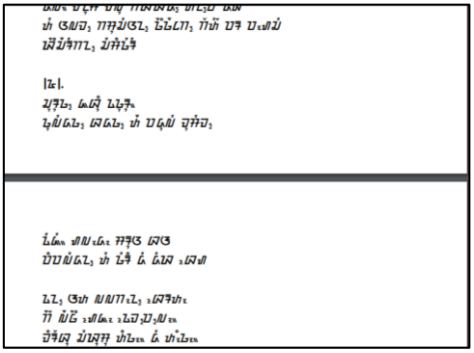


Gambar 5.11 Antarmuka Modul Embedding Skenario C

Keluaran proses *embedding* adalah stego.pdf. Tampilan stego.pdf ditunjukkan pada Gambar 5.13. Secara isi teks, hasil stego.pdf tidak jauh berbeda dengan masukan Walungan.docx yang ditunjukkan pada Gambar 5.12. Namun, secara tata letak, terdapat perbedaan pada pemenggalan baris antar halaman. Kedua berkas, Walungan.docx maupun stego.pdf memiliki dua halaman.



Gambar 5.12 Masukan docx Skenario C



Gambar 5.13 Keluaran stego.pdf Skenario C

Perbandingan ditunjukkan pada gambar 5.14. Perbandingan tersebut diambil dari baris pertama cover text dan stego text. Karena proses embedding, teks dari stego.pdf mengalami pergeseran pada beberapa rarangkén. Contoh rarangkén yang mengalami pergeseran adalah rarangkén ke-1, Pergeseran tersebut sesuai dengan 2 bit pertama dari bit *secret message* yaitu 10, Bit 0 menunjukan tidak ada pergeseran yang akan dilakukan, sedangkan bit 1 menunjukan ada pergeseran yang akan dilakukan.

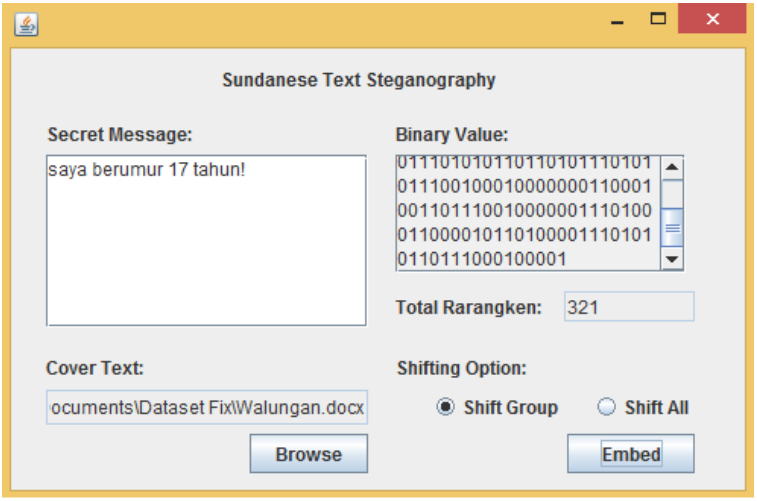
Gambar 5.14 Perbandingan Cover (Atas) dan Stego (Bawah) Skenario C

Hasil modul *extracting* pada Skenario C ditunjukkan pada Gambar 5.15. Dengan menggunakan *shifting option* yang sama dengan modul *embedding* yaitu *shift all*, hasil keluaran *secret message* adalah "saya berumur 17 tahun!". *Secret message* ini sama dengan *secret message* yang dimasukan pada modul *embedding*.

Gambar 5.15 Antarmuka Modul Extracting Skenario C

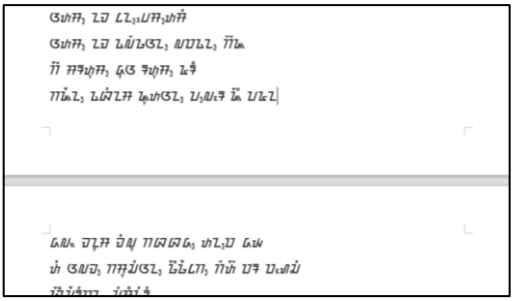
5.4.1.4 Hasil Uji Coba Fungsionalitas Skenario D

Hasil uji coba fungsionalitas Skenario D pada proses *embedding* ditunjukkan pada Gambar 5.16. Setelah mengisi data masukan sesuai skenario, sistem memunculkan hasil *binary value* dari *secret message*. Selain itu, terdapat total rarangkén sesuai *shifting option* yang dipilih. Pada skenario D, data ke-4 memiliki 321 rarangkén jika dihitung menggunakan pilihan *shift group.* Hasil ini lebih tinggi dibandingkan hasil pada skenario C.

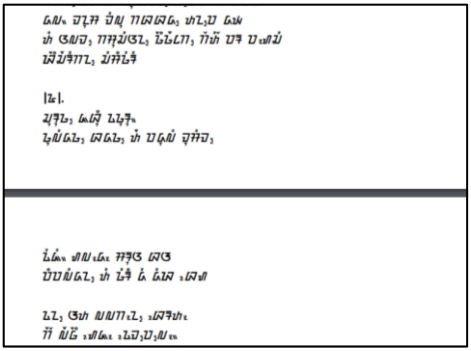


Gambar 5.16 Antarmuka Modul Embedding Skenario D

Keluaran proses *embedding* adalah stego.pdf. Tampilan stego.pdf ditunjukkan pada Gambar 5.18. Secara isi teks, hasil stego.pdf tidak jauh berbeda dengan masukan Walungan.docx yang ditunjukkan pada Gambar 5.17. Namun, secara tata letak, terdapat perbedaan pada pemenggalan baris antar halaman. Kedua berkas, Walungan.docx maupun stego.pdf memiliki dua halaman.



Gambar 5.17 Masukan docx Skenario D



Gambar 5.18 Keluaran stego.pdf Skenario D

Perbandingan ditunjukkan pada gambar 5.19. Perbandingan tersebut diambil dari baris pertama *cover text* dan *stego text*. Karena proses *embedding*, teks dari stego.pdf mengalami pergeseran pada beberapa rarangkén. Contoh rarangkén yang mengalami pergeseran adalah rarangkén ke-1, Pergeseran tersebut sesuai dengan 2 bit pertama dari bit *secret message* yaitu 10, Pergeseran yang dilakukan pada 2 bit pertama ssama dengan pergeseran pada Skenario C karena tidak terdapat rarangkén yang berpasangan.

Gambar 5.19 Perbandingan Cover (Atas) dan Stego (Bawah) Skenario D

Hasil modul *extracting* pada Skenario B ditunjukkan pada Gambar 5.10, Dengan menggunakan *shifting option* yang sama dengan modul *embedding* yaitu *shift group*, hasil keluaran *secret message* adalah "saya rijal". *Secret message* ini sama dengan *secret message* yang dimasukan pada modul *embedding.*

Gambar 5.20 Antarmuka Modul Extracting Skenario

5.4.2 Hasil Uji Coba Panjang Secret Message

Uji coba panjang *secret message* dilakukan dengan menghitung minimal rarangkén yang diperlukan *cover text* untuk menyisipkan satu karakter, maksimal karakter yang dapat disisipkan dan jumlah rarangkén yang diperlukan *cover text* untuk mendapatkan nilai maksimal karakter. Hasil penghitungan ditunjukkan pada Tabel 5.4.

Tabel 5.4 Penanda Panjang Secret Message

Skenario	Min. Rarangkén	Maks. Karakter	Jml. Rarangkén
A	16	32	264
B	17	64	521
C	18	128	1034
D	19	256	2059
E	20	512	4108

Minimal rarangkén yang dibutuhkan *cover text* untuk menyisipkan satu karakter dihitung berdasarkan jumlah bit penanda panjang *secret message* ditambah dengan 8 bit yang merepresentasikan satu karakter. Pada skenario A, bit penanda panjang *secret message* yang berjumlah 8 bit ditambahkan 8 bit representasi satu karakter sehingga minimal rarangkén berjumlah 16 bit. Sementara itu, Maksimal karakter dihitung berdasarkan 2^n dibagi 8 dimana n adalah penanda panjang *secret message.* Jumlah rarangkén yang dibutuhkan *cover text* untuk menampung maksimal karakter dihitung berdasarkan 2^n + n dimana n adalah penanda panjang *secret message*.

Setiap penambahan satu bit penanda panjang secret message, terdapat kenaikan 2 kali maksimal karakter yang dapat disisipkan. Semakin tinggi bit penanda panjang *secret message*, maka semakin banyak karakter yang bisa disisipkan dan semakin banyak jumlah rarangkén yang harus dimiliki oleh *cover text.*

Tabel 5.5 Total Rarangkén dan Panjang Karakter

Data	Total Rarangkén		Menampung Maksimal				
	Shift All	Shift Group	A	B	C	D	E
1	187	193	X	X	X	X	X
2	188	206	X	X	X	X	X
3	**158**	168	X	X	X	X	X
4	307	321	V	X	X	X	X
5	244	254	X	X	X	X	X
6	1300	1372	V	V	V	X	X
7	3281	**3562**	V	V	V	V	X
8	408	434	V	X	X	X	X
9	344	360	V	X	X	X	X
10	1572	1671	V	V	V	X	X

Hasil penghitungan pada Tabel 5.4 disesuaikan dengan total rarangkén yang dimiliki setiap data. Pada Tabel 5.5 ditunjukkan data yang dapat menampung karakter maksimal dari setiap skenario. Tanda "X" diberikan jika data tersebut tidak dapat menampung karakter maksimal dari skenario yang ada. Tanda "V" diberikan jika data tersebut dapat menampung karakter maksimal dari skenario yang ada.

Pada perbandingan yang dilakukan, data ke-4, 8, dan 9 dapat menampung maksimal karakter dari skenario A yaitu 32 karakter. Sementara itu, data ke-6, 7 dan 10 dapat menampung maksimal karakter dari skenario A, B, dan C atau sejumlah 128 karakter. Data ke-7 yang mempunyai rarangkén tertinggi dapat menampung maksimal karakter dari skenario D yaitu 256 karakter. Data ke-3 merupakan data dengan total rarangkén terendah.

5.4.3 Hasil Uji Coba Perubahan Ukuran

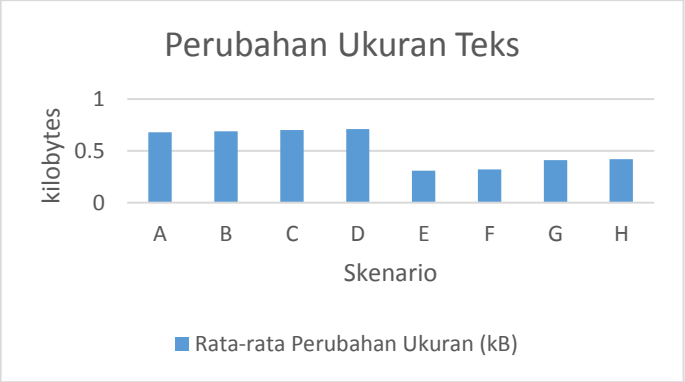
Uji coba perubahan ukuran dihitung dengan membandingkan ukuran *cover text* dan *stego text* yang dihasilkan untuk setiap skenario. Hasil uji coba ditunjukkan pada Tabel 5.6 dan Gambar 5.21. Untuk skenario A, B, C, dan D dilakukan 3 kali percobaan karena skenario tersebut menggunakan algoritma pergeseran acak sisa rarangkén. Hasil 3 kali percobaan tersebut

kemudian didapatkan rata-ratanya. Skenario E, F, G dan H hanya dilakukan 1 kali percobaan karena menghasilkan nilai yang tetap.

Tabel 5.6 Perubahan Ukuran Teks Semua Skenario

Skenario	Rata-rata (kB)		Persentase Perubahan
	Perubahan Ukuran	Total	
A	0,68	0,70	47,14%
B	0,69		
C	0,70		
D	0,71		
E	0,31	0,37	
F	0,32		
G	0,41		
H	0,42		

Secara umum, skenario yang tidak menggunakan algoritma pergeseran acak sisa rarangkén yaitu skenario E, F, G, dan H memiliki perubahan ukuran yang lebih kecil yaitu 0,37 kiloByte. Sementara itu skenario A, B, C, dan D yang menggunakan algoritma pergeseran acak sisa rarangkén, memiliki perubahan sebesar 0,70 kiloByte. Terdapat penurunan sebesar 47,14% jika algoritma pergeseran acak sisa rarangkén dimatikan.



Gambar 5.21 Grafik Perubahan Ukuran Teks

Dari uji coba yang dilakukan, pilihan *shifting option* tidak terlalu berpengaruh pada perubahan ukuran pada stego text yang dihasilkan. Masukan *secret message* pada uji coba ini secara umum berpengaruh pada perubahan ukuran teks. Untuk mengetahui pengaruh ini dapat dilihat melalui perbandingan skenario E, F, G, dan H yang tidak menggunakan algoritma pergeseran acak sisa rarangkén pada Tabel 5.7. Pemilihan skenario E, F, G, dan H dilakukan agar nilai perubahan yang diberikan tetap (konstan). Pada skenario E dan F, terdapat perubahan teks sebesar 0,32 kiloByte untuk penyisipan 5 karakter *secret message*. Sementara itu pada skenario G dan H, terdapat perubahan teks sebesar 0,41 kiloByte untuk penyisipan 15 karakter *secret message*. Terdapat kenaikan sebesar 31,25% untuk penambahan 10 karakter.

Tabel 5.7 Perubahan Ukuran Teks Skenario E, F, G, dan H

Skenario	Rata-rata (kB)		Persentase Perubahan
	Perubahan Ukuran	Total	
E	0,31	0,32	31,25%
F	0,32		
G	0,41	0,42	
H	0,42		

5.4.4 Hasil Uji Coba Waktu

Hasil uji coba waktu berisi tentang waktu yang dibutuhkan untuk proses *embedding* dan *extracting*. Setiap data, dilakukan percobaan sebanyak 3 kali kemudian didapatkan rata-ratanya.

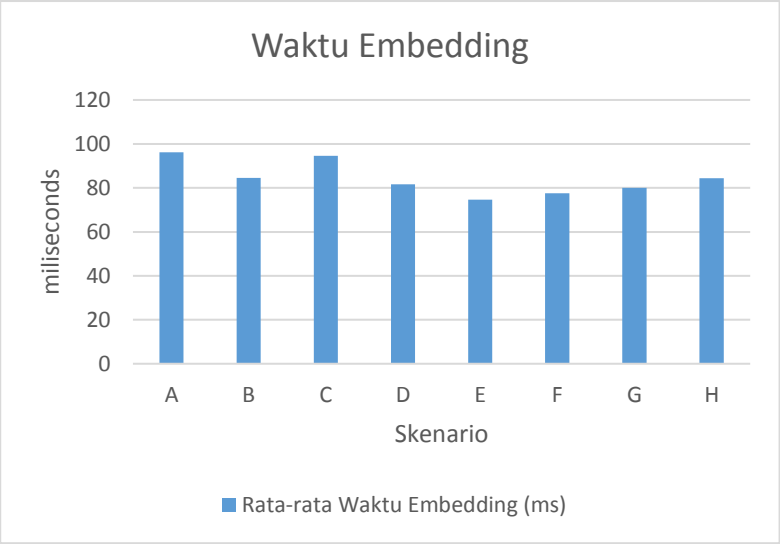
5.4.4.1 Hasil Uji Coba Waktu Embedding

Uji coba waktu embedding dihitung ketika proses pergeseran rarangkén dijalankan. Hasil uji coba ditunjukkan pada Tabel 5.8 dan Gambar 5.22. Secara umum, skenario A, B, C, dan D yang menggunakan algoritma pergeseran acak sisa rarangkén memiliki waktu *embedding* 89,24 *miliseconds*. Sementara itu, skenario E, F, G dan H yang tidak menggunakan pergeseran acak

memiliki waktu *embedding* yang lebih kecil yaitu 79,13 *miliseconds*. Terdapat penurunan 11,32% ketika algoritma pergeseran acak sisa rarangkén dimatikan.

Tabel 5.8 Waktu Embedding Semua Skenario

Skenario	Rata-rata (ms)		Persentase Perubahan
	Waktu Embedding	Total	
A	96,23	89,24	11,32%
B	84,49		
C	94,55		
D	81,70		
E	74,57	79,14	
F	77,52		
G	80,00		
H	84,46		



Gambar 5.22 Grafik Waktu Embedding

Masukan shifting option pada skenario A, B, C, dan D yang menggunakan algoritma pergeseran acak sisa rarangkén (keadaan maksimum) berpengaruh pada waktu *embedding secret message* seperti ditunjukkan pada Tabel 5.9. Secara umum, skenario A dan C dengan pilihan *shift all* memiliki waktu embedding 95,28 *miliseconds*. Sementara itu, skenario B dan D dengan pilihan *shift group* memiliki waktu embedding 82,96 *miliseconds*. Terjadi penurunan sebesar 12,93%. Hal ini disebabkan karena implementasi algoritma *shift all* memiliki proses yang lebih kompleks.

Tabel 5.9 Waktu Embedding Skenario A, C, B, dan D

Skenario	Rata-rata (ms)		Persentase Perubahan
	Waktu Embedding	Total	
A	96,23	95,28	12,93%
C	94,55		
B	84,49	82,96	
D	81,70		

Waktu *embedding* skenario E, F, G dan H yang tidak menggunakan algoritma pergeseran acak sisa rarangkén dipengaruhi oleh panjang masukan *secret message*. Besar waktu *embedding* naik 8,13% dari 76,05 *milisecods* pada skenario E dan F yang memiliki 5 karakter, menjadi 82,23 *miliseconds* pada skenario G dan H yang memiliki 15 karakter. Detail waktu ditunjukkan pada Tabel 5.10.

Tabel 5.10 Waktu Embedding Skenario E, F, G, dan H

Skenario	Rata-rata (ms)		Persentase Perubahan
	Waktu Embedding	Total	
E	74,57	76,05	8,13%
F	77,52		
G	80,00	82,23	
H	84,46		

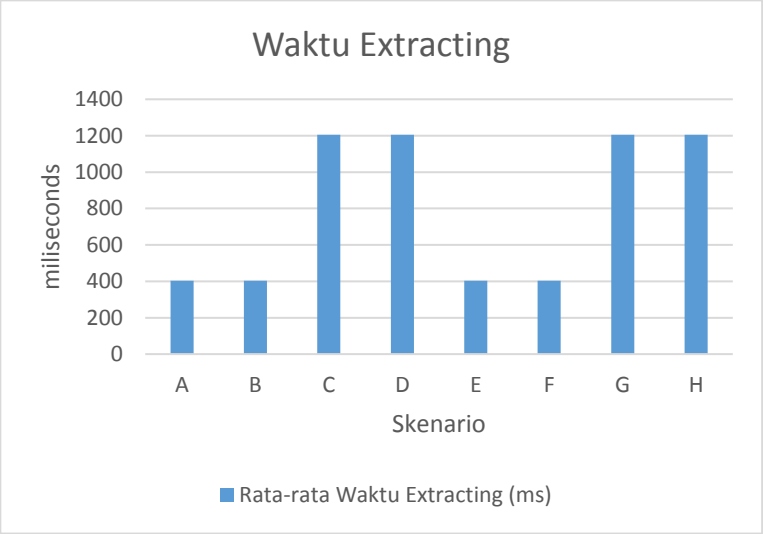
5.4.4.2 Hasil Uji Coba Waktu Extracting

Uji coba waktu *extracting* dihitung ketika proses ekstraksi informasi dijalankan. Hasil uji coba ditunjukkan pada Tabel 5.11 dan Gambar 5.23. Untuk mempermudah penghitungan, urutan skenario pada tabel diurutkan berbeda menjadi A, B, E, F, C, D, G dan H.

Tabel 5.11 Waktu Extracting Semua Skenario

Skenario	Rata-rata (ms)		Persentase Perubahan
	Waktu Extracting	Total	
A	402,87	402,87	198,99%
B	402,71		
E	402,99		
F	402,90		
C	1204,72	1204,54	
D	1204,33		
G	1204,68		
H	1204,41		

Secara umum, skenario A, B, E dan F yang memiliki masukan 5 karakter *secret message* membutuhkan waktu 402,87 *miliseconds*, sedangkan C, D, G, dan H yang memiliki masukan 15 karakter *secret message* membutuhkan waktu 1204,54 *miliseconds*. Terdapat kenaikan 198,99% jika karakter yang menjadi masukan bertambah 10 karakter *secret message*. Tidak ada perbedaan signifikan pada penggunaan algoritma pergeseran acak sisa rarangkén atau pemilihan *shifting option*. Hal ini disebabkan oleh algoritma ekstraksi yang berhenti ketika bit informasi sudah sesuai dengan penanda panjang yang ada.



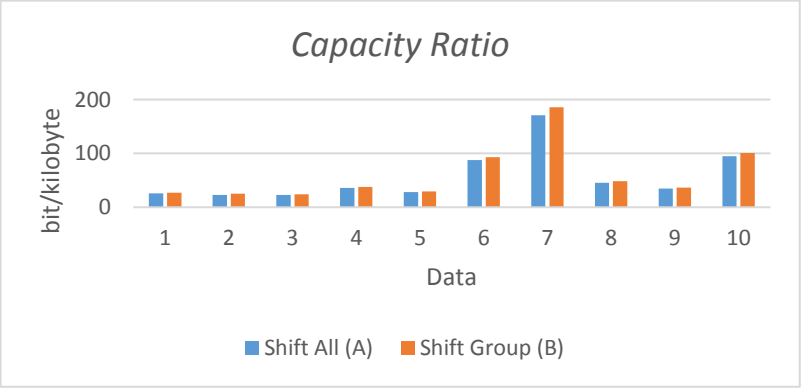
Gambar 5.23 Grafik Waktu Extracting

5.4.5 Hasil Uji Coba Capacity Ratio

Uji coba *capacity ratio* dihitung dengan membagi total rarangkén dengan ukuran *cover text*. Hasil penghitungan ditunjukan pada Tabel 5.11 Besaran hasil *capacity ratio* adalah *bit/kiloByte*. Dari uji coba yang dilakukan, data ke-6, 7 dan 10 memiliki *capacity ratio* yang lebih tinggi dibandingkan data lainnya. Perbedaan yang signifikan dari ketiga data tersebut dengan yang lainnya adalah total rarangkén yang nilainya mencapai lebih dari 1000 rarangkén.

Tabel 5.12 Hasil Uji Coba Capacity Ratio

Data	Total Rarangkén (bit)		Ukuran Cover Text (kB)	Capacity Ratio (b/kB)	
	All	Group		A	B
1	187	193	7,17	26,08	26,91
2	188	206	8,24	22,82	25,00
3	158	168	6,92	22,83	24,28
4	307	321	8,48	36,20	37,85
5	244	254	8,59	28,41	29,57
6	1300	1372	14,78	87,96	92,83
7	3281	3562	19,20	170,89	185,52
8	408	434	8,96	45,54	48,44
9	344	360	9,88	34,82	36,44
10	1572	1671	16,62	94,58	100,54
			Rata-rata	57,01	60,74
			Persentase Perubahan		6,54%



Gambar 5.24 Grafik Capacity Ratio

Secara umum, skenario B yang menggunakan *shift group* memiliki *capacity ratio* yang lebih tinggi disebabkan total rarangkén yang lebih banyak. Terdapat kenaikan 6,54% dari semula 57,01 *bit/kiloByte* pada skenario A dengan *shift all* menjadi 60,74 *bit/kiloByte* pada skenario B dengan *shift group*. Grafik *capacity ratio* ditunjukkan pada Gambar 5.24.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah. Selain itu juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari uji coba dan evaluasi sistem steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding* adalah sebagai berikut:

1. Steganografi teks pada aksara Sunda dengan pendekatan *Feature Coding* pada proses *embedding* dilakukan dengan menggeser rarangkén sesuai pilihan pergeseran (*shifting option*) yang ada. Pergeseran dilakukan dengan mengganti blok Unicode rarangkén normal menjadi blok Unicode rarangkén modifikasi. Pada proses *extracting* dilakukan dengan mendeteksi rarangkén yang telah bergeser untuk didapatkan informasi rahasia.
2. Maksimal karakter yang dapat disisipkan pada *cover text* dipengaruhi oleh bit penanda panjang *secret message* dan kapasitas *cover text*. Ukuran *stego text* yang dihasilkan dipengaruhi oleh masukan *secret message* dan algoritma pengacakan sisa rarangkén. Terdapat kenaikan sebesar 31,25% untuk perbedaan 10 karakter masukan *secret message*. *Stego text* tanpa algoritma pengacakan sisa rarangkén turun 47,14%. Waktu *embedding* secara umum dipengaruhi oleh penggunaan algoritma pergeseran acak sisa rarangkén. Terdapat penurunan 11,32% jika algoritma pergeseran acak sisa rarangkén tidak diaktifkan. Waktu *extracting* dipengaruhi oleh panjang *masukan secret message*. Terdapat kenaikan 198,99% untuk perbedaan 10 karakter masukan secret message.

3. Capacity Ratio dari sebuah *cover text* dalam aksara Sunda dipengaruhi oleh jumlah rarangkén yang terdapat pada *cover text* tersebut. Pilihan *shift all* menghasilkan rata-rata *capacity ratio* 57,01 *bit/kiloByte*. Pilihan *shift group* menghasilkan rata-rata *capacity ratio* yang lebih besar yaitu 60,74 *bit/kiloByte*. Terdapat kenaikan sebesar 6,54%.

6.2 Saran

Saran yang dapat diberikan terkait pengembangan tugas akhir ini adalah sebagai berikut:

1. Perbaikan tata letak dalam PDF yang dihasilkan agar berkas *stego text* dan *cover text* tidak mempunyai perbedaan yang signifikan.
2. Proteksi PDF dengan *password* untuk meningkatkan ketahanan dari *stego text*.
3. Penggunaan *library* berbayar yang mendukung OpenType Layout pada *font* untuk menghasilkan aksara yang sesuai.

DAFTAR PUSTAKA

[1] S. Changder, D. Ghosh and N. C. Debnath, "Linguistic approach for text steganography through Indian text," in *2nd Int. Conf. on Computer Technology and Development*, 2010.

[2] D. Kahn, The Code Breakers- the comprehensive history of secret, Schribner, 1996.

[3] J. T. Brassil, S. Low, N. F. Maxemchuk and O. L, "Electronic Marking and Identification Techniques to Discourage Document Copying," *IEEE Journal on Selected Areas in Communications,* vol. 13, no. 8, pp. 1495-1504, 1995.

[4] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new approach to persian/arabic text steganography," in *5th Int. Conf. Computer and Information Science*, Washington, 2006.

[5] A. Gutub and M. Fattani, "A Novel Arabic Text Steganography Method," in *WASET International Conference on Computer,*, Vienna, 2007.

[6] M. Aabed, S. Awaideh, A. Elshafei and A. Gutub, "Arabic Diacritics Based Steganograophy," in *IEEE International Conference on Signal Processing and Communications (ICSPC'07)*, 2007.

[7] A. Odeh, K. Elleithy and M. Faezipour, "Steganography in Arabic text using Kashida variation algorithm (KVA)," in *Systems, Applications and Technology Conference (LISAT)*, Long Island, 2013.

[8] A. Odeh, A. Alzubi, Q. B. Hani and K. Elleithy, "Steganography by multipoint Arabic letters," in *Systems, Applications and Technology Conference (LISAT)*, Long Island, 2012.

[9] K. Alla and R. S. R. Prasad, "A Novel Hindi Text Steganography using Diacritics and its Compound words," *International Journal of Computer Science and Network Security (IJCSN)*, vol. 8, no. 12, pp. 404-408, 2008.

[10] S. Changder, N. C. Debnath and D. Ghosh, "Hindi Text Steganography by Shifting of Matra," in *International Conference on Advances in Recent Technologies in Communication and Computing*, 2009.

[11] K. Alla and R. S. R. Prasad, "A New Approach to Hindi Text Steganography Using Matraye, Core Classification And HHK Scheme," in *Seventh International Conference on Information Technology*, 2010.

[12] K. Alla and R. S. R. Prasad, "A new approach to Telugu text steganography," in *IEEE Symposium on Wireless Technology and Applications (ISWTA)*, 2011.

[13] Tim Unicode Aksara Sunda 2008, Direktori Aksara Sunda untuk Unicode, Pemerintah Provinsi Jawa Barat, 2008.

[14] S. Singh, The Code Book: The Science of Secrecy from Ancient Egypt to Quantum, New York: Anchor Books, 1999.

[15] S. Sharma, A. Gupta, M. C. Trivedi and V. K. Yadav, "Analysis of Different Text Steganography Techniques : A Survey," in *Second International Conference on Computational Intelligence & Communication Technology*, 2016.

[16] S. Kingslin and N. Kavitha, "Evaluative Approach towards Text Steganographic Techniques," *Indian Journal of Science and Technology*, vol. 8, no. 29, 2015.

[17] K. Rabah, “Steganography-The Art of Hiding Data,” *Information Technology Journal*, vol. 3, no. 3, pp. 245-269, 2004.

[18] J. Gosling, B. Joy, G. Steele and G. Bracha, The Java Language Specification, Second Edition ed., Mountain View, California: Sun Microsystem, 2000.

[19] A. Kadir, Buku Pertama Belajar Pemrograman Java Untuk Pemula, Yogyakarta: Mediakom, 2014.

[20] Sun Microsystems, “NetBeans IDE - Overview,” Sun Microsystems, [Online]. Available: https://netbeans.org/features/index.html.

[21] Apache, “Apache PDFBox,” Apache, [Online]. Available: https://pdfbox.apache.org/.

[22] G. Williams, “Font Creation with FonForge,” 2003. [Online]. Available: http://root.tug.org/TUGboat/tb24-3/williams.pdf.

[23] Tutorials Point, “Apache POI Tutorial,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/apache_poi/.

[24] Y. Hendayana, “Sajak Bahasa Sunda (Halodo Teuing Ku Panjang),” bahasa-sunda.com, [Online]. Available: http://bahasa-sunda.com/contoh-sunda/sajak-bahasa-sunda/sajak-bahasa-sunda-halodo-teuing-ku-panjang.html.

[25] I. Nugraha, “Manglé - Majalah Mingguan Basa Sunda | Itungan Samémèh Balitungan,” Manglé, [Online]. Available: http://mangle-online.com/pidangan/rubrik/sjk/Manglé%20-%20Majalah%20Mingguan%20Basa%20Sunda%20_%20Itungan%20Samémèh%20Balitungan.htm.

[26] Risnawati, “Manglé - Majalah Mingguan Basa Sunda | Sekar Merdika,” Manglé, [Online]. Available:

http://mangle-online.com/pidangan/rubrik/sjk/Manglé%20-%20Majalah%20Mingguan%20Basa%20Sunda%20_%20Sekar%20Merdika.htm.

[27] U. Romli HM, "Manglé - Majalah Mingguan Basa Sunda | Walungan," Manglé, [Online]. Available: http://mangle-online.com/pidangan/rubrik/sjk/Manglé%20-%20Majalah%20Mingguan%20Basa%20Sunda%20_%20Walungan.htm.

[28] C. Nurzaman, "Manglé - Majalah Mingguan Basa Sunda | Baris Kahiji," Manglé, [Online]. Available: http://mangle-online.com/pidangan/rubrik/sjk/Manglé%20-%20Majalah%20Mingguan%20Basa%20Sunda%20_%20Baris%20Kahiji.htm.

[29] A. Karsono and C. Retty Isnendes, Lir Cahya Nyorot Eunteung, Bandung: Sonagar Press, 2008.

[30] G. Suwarna, "Kumpulan Sajak-Sajak Godi Suwarna," Cikancah Cyber, [Online]. Available: http://www.cikancah-cyber.com/2016/09/kumpulan-sajak-sajak-godi-suwarna.html.

[31] A. Rosidi, "Sajak Bahasa Sunda (Jante Arkidam)," bahasa-sunda.com, [Online]. Available: https://bahasa-sunda.com/contoh-sunda/sajak-bahasa-sunda/sajak-bahasa-sunda-jante-arkidam.html.

[32] L. Taliah, "Sajak-sajak Lia Taliah," sundanews.com, [Online]. Available: http://www.sundanews.com/aos/seratan/1352485842.

[33] SMA Pasundan 2 Tasikmalaya, *ANTOLOGI PASANGGIRI MACA SAJAK SUNDA XIV,* Tasikmalaya: SMA Pasundan 2 Tasikmalaya, 2015.

[34] Adobe, “What is PDF? Adobe Portable Document Format | Adobe Acrobat DC,” Adobe, [Online]. Available: https://acrobat.adobe.com/us/en/why-adobe/about-adobe-pdf.html.

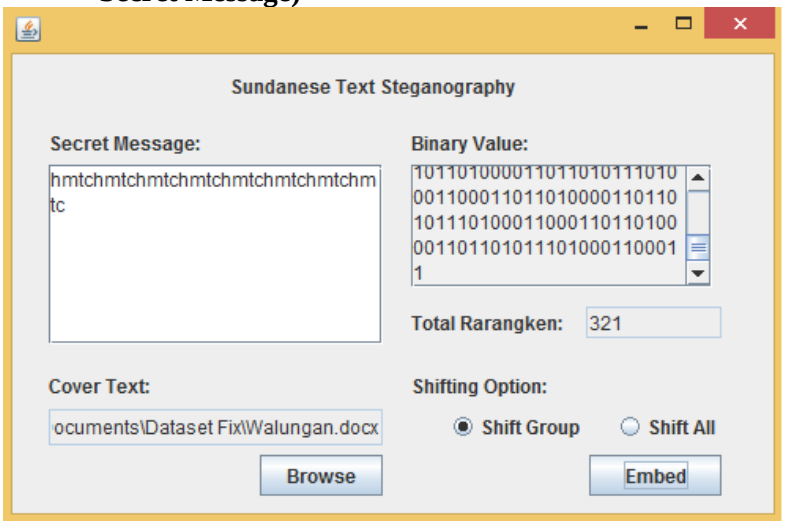
[*Halaman ini sengaja dikosongkan*]

LAMPIRAN

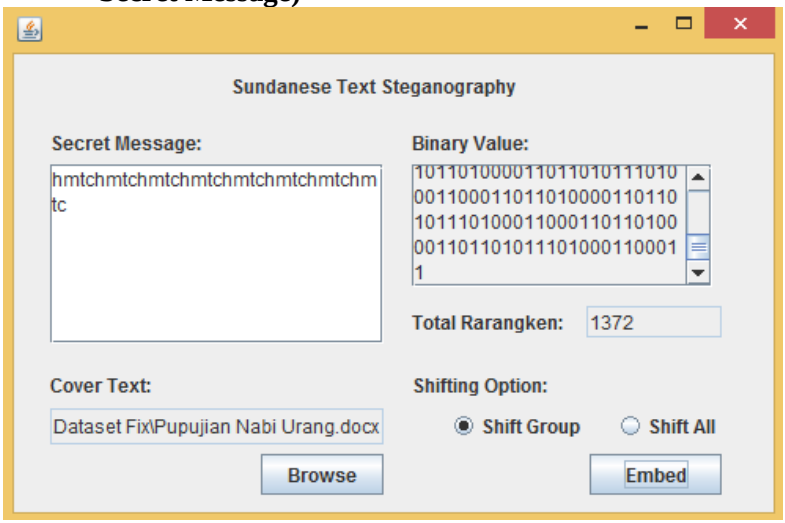
A.1. Masukan Uji Coba Panjang Secret Message

Skenario	Maks. Karakter	Teks
A	32	hmtchmtchmtchmtchmtchmtchmtchmtc
B	64	hmtchmtchmtchmtchmtchmtchmtchmtchm tchmtchmtchmtchmtchmtchmtchmtc
C	128	hmtchmtchmtchmtchmtchmtchmtchmtchm tchmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtc
D	256	hmtchmtchmtchmtchmtchmtchmtchmtchm tchmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtc
E	512	hmtchmtchmtchmtchmtchmtchmtchmtchm tchmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt chmtchmtchmtchmtchmtchmtchmtchmtch mtchmtchmtchmtchmtchmtchmtchmtchmt c

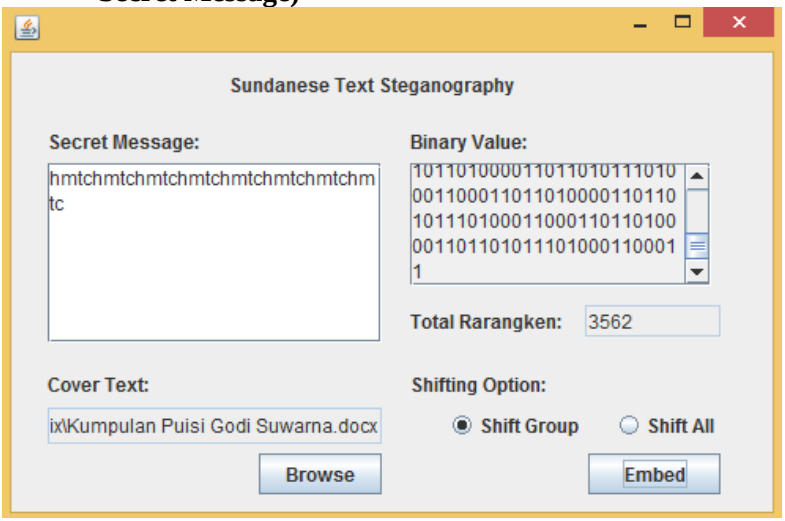
A.2. Hasil Uji Skenario A pada Data 4 (Uji Coba Panjang Secret Message)



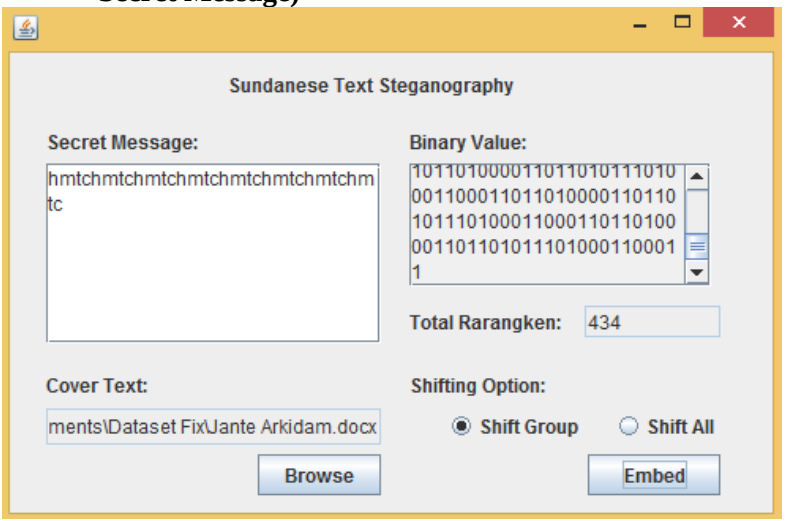
A.3. Hasil Uji Skenario A pada Data 6 (Uji Coba Panjang Secret Message)



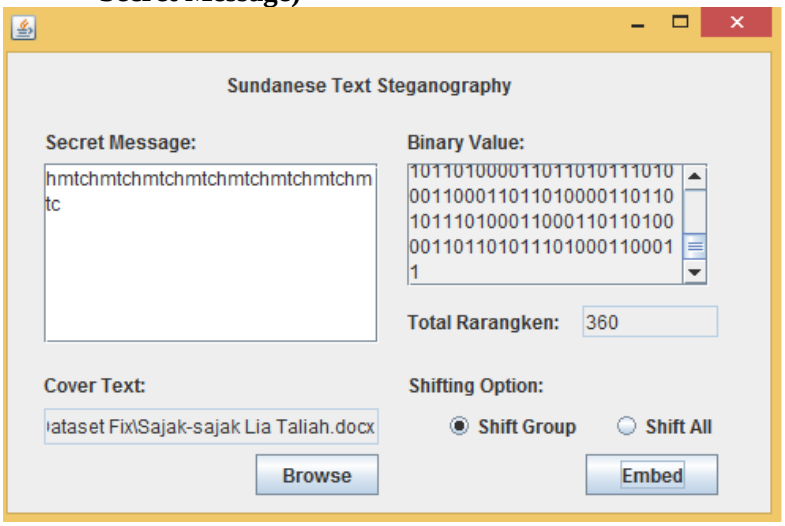
A.4. Hasil Uji Skenario A pada Data 7 (Uji Coba Panjang Secret Message)



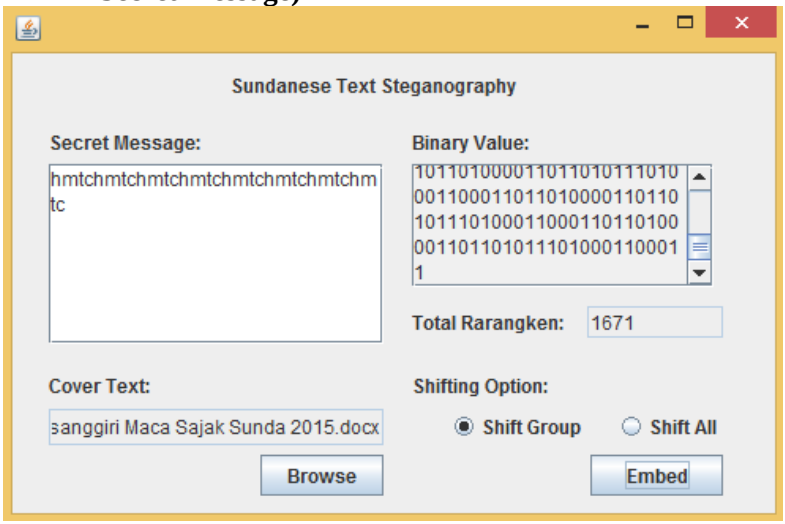
A.5. Hasil Uji Skenario A pada Data 8 (Uji Coba Panjang Secret Message)



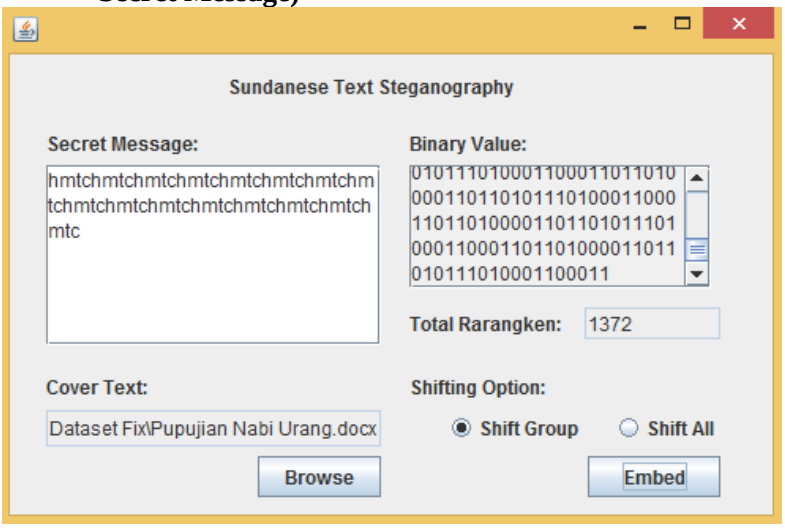
A.6. Hasil Uji Skenario A pada Data 9 (Uji Coba Panjang Secret Message)



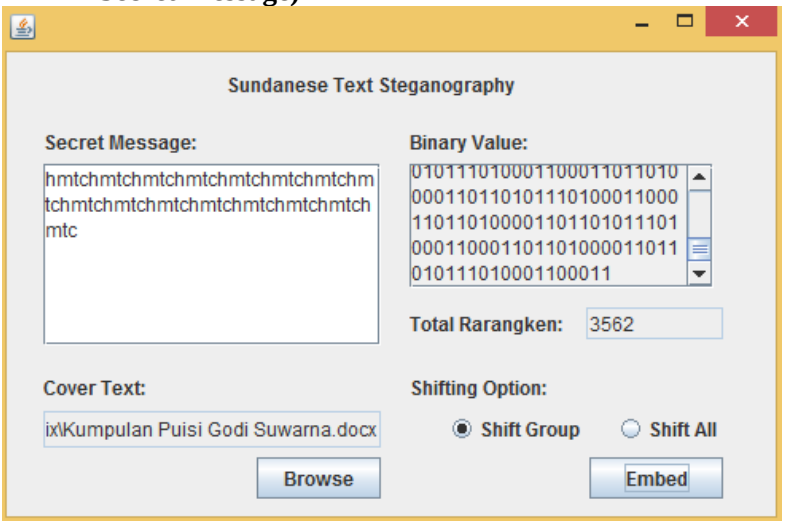
A.7. Hasil Uji Skenario A pada Data 10 (Uji Coba Panjang Secret Message)



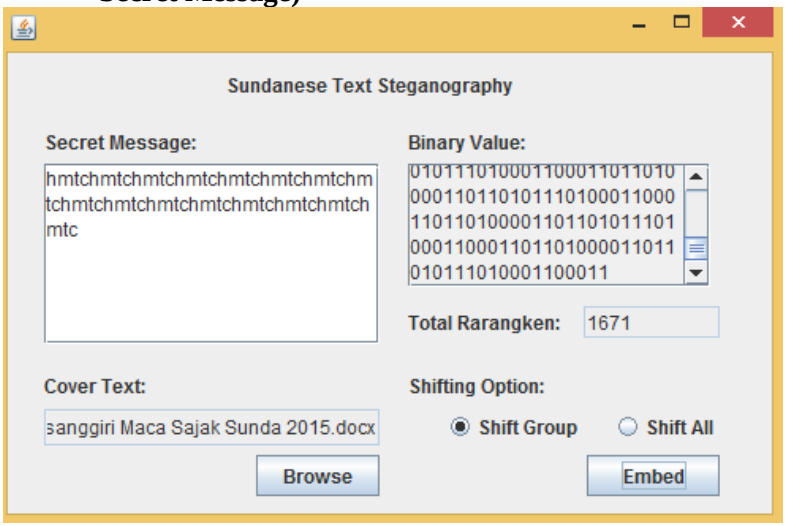
A.8. Hasil Uji Skenario B pada Data 6 (Uji Coba Panjang Secret Message)



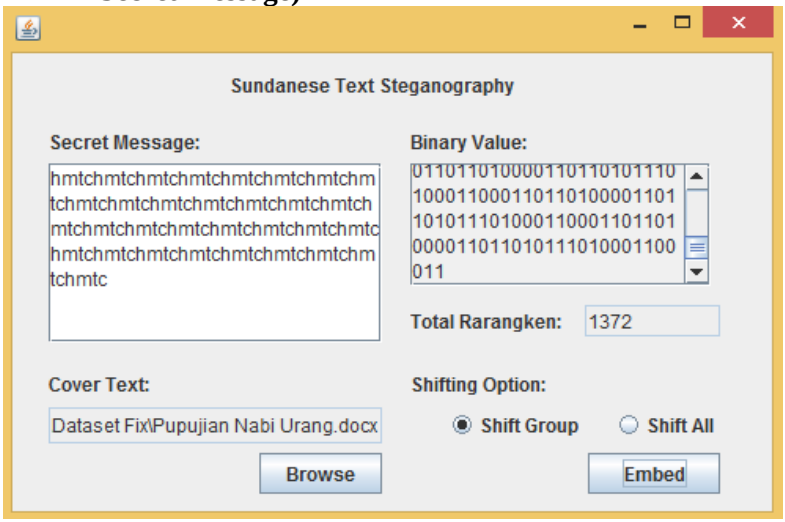
A.9. Hasil Uji Skenario B pada Data 7 (Uji Coba Panjang Secret Message)



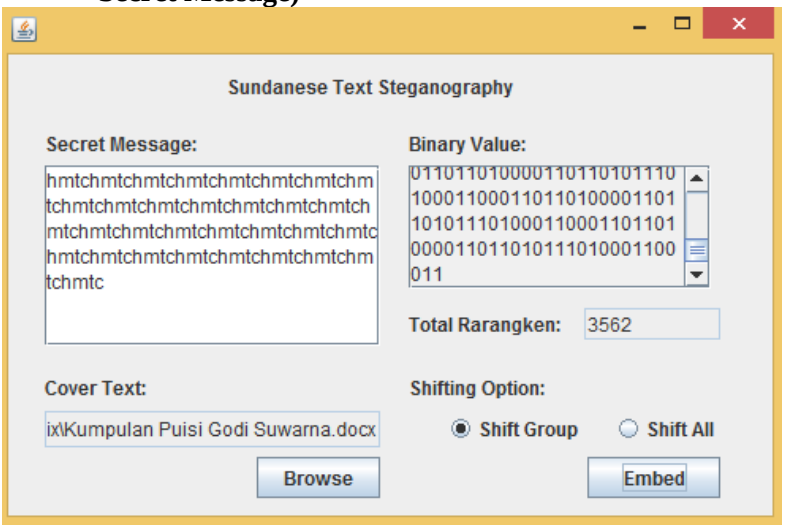
A.10. Hasil Uji Skenario B pada Data 10 (Uji Coba Panjang Secret Message)



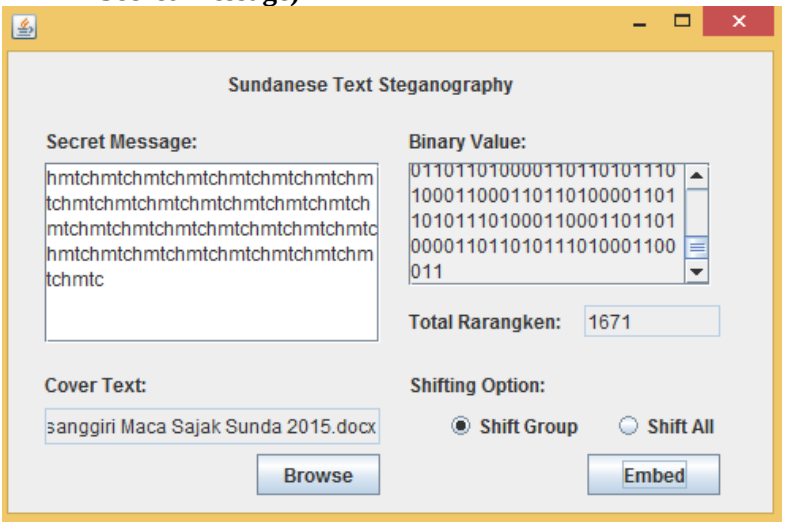
A.11. Hasil Uji Skenario C pada Data 6 (Uji Coba Panjang Secret Message)



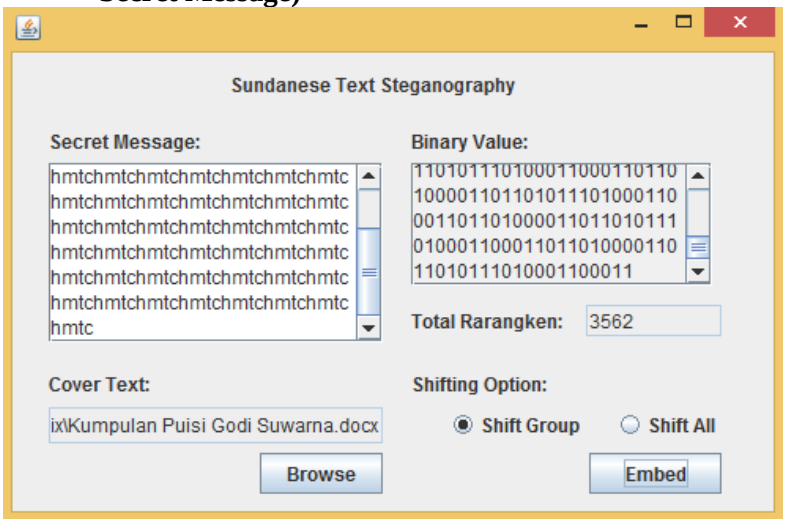
A.12. Hasil Uji Skenario C pada Data 7 (Uji Coba Panjang Secret Message)



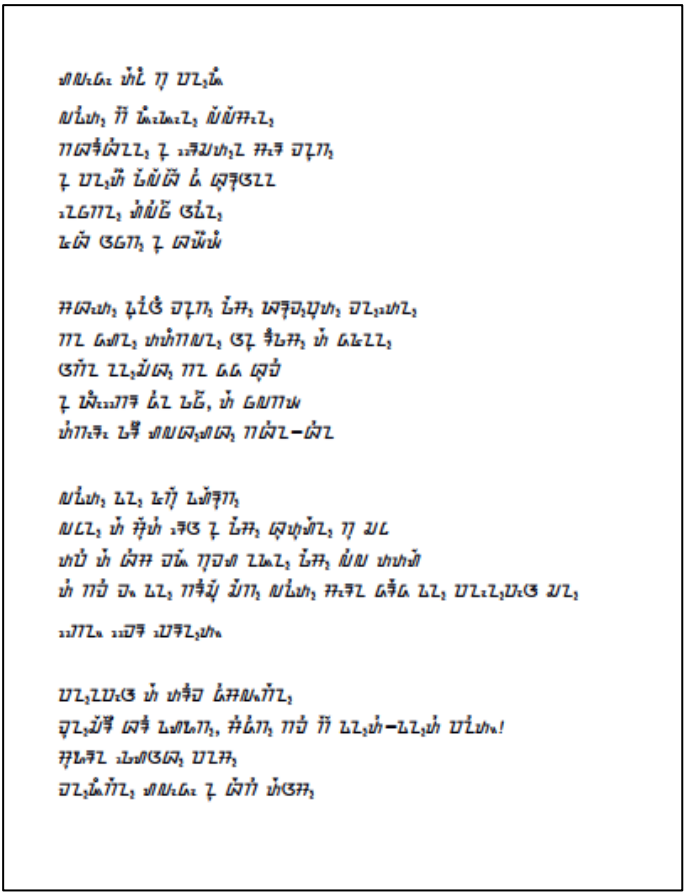
A.13. Hasil Uji Skenario C pada Data 10 (Uji Coba Panjang Secret Message)



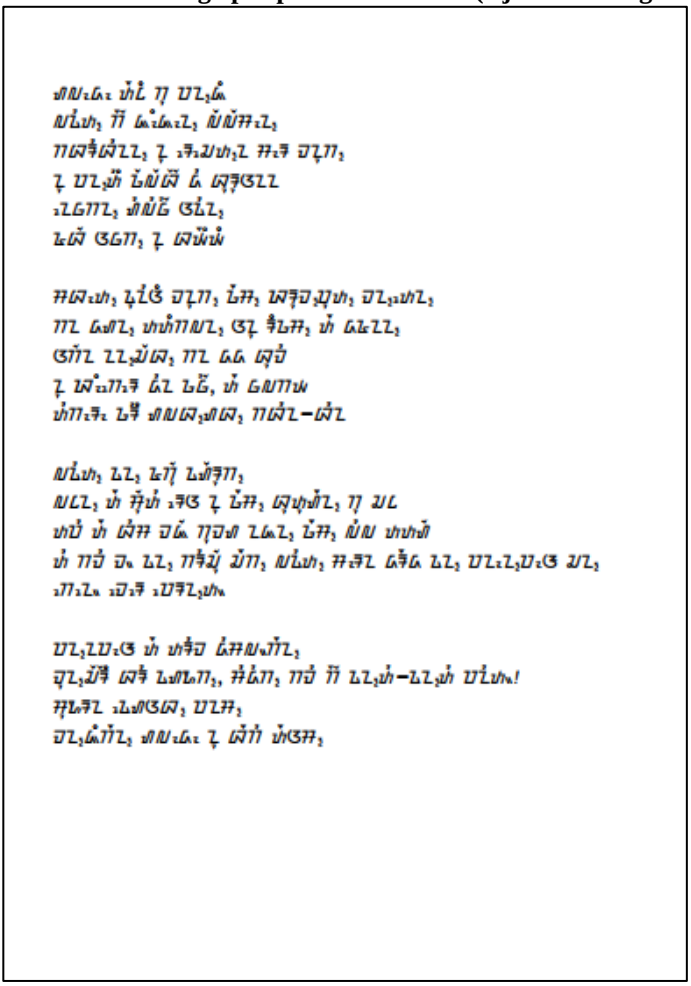
A.14. Hasil Uji Skenario D pada Data 7 (Uji Coba Panjang Secret Message)



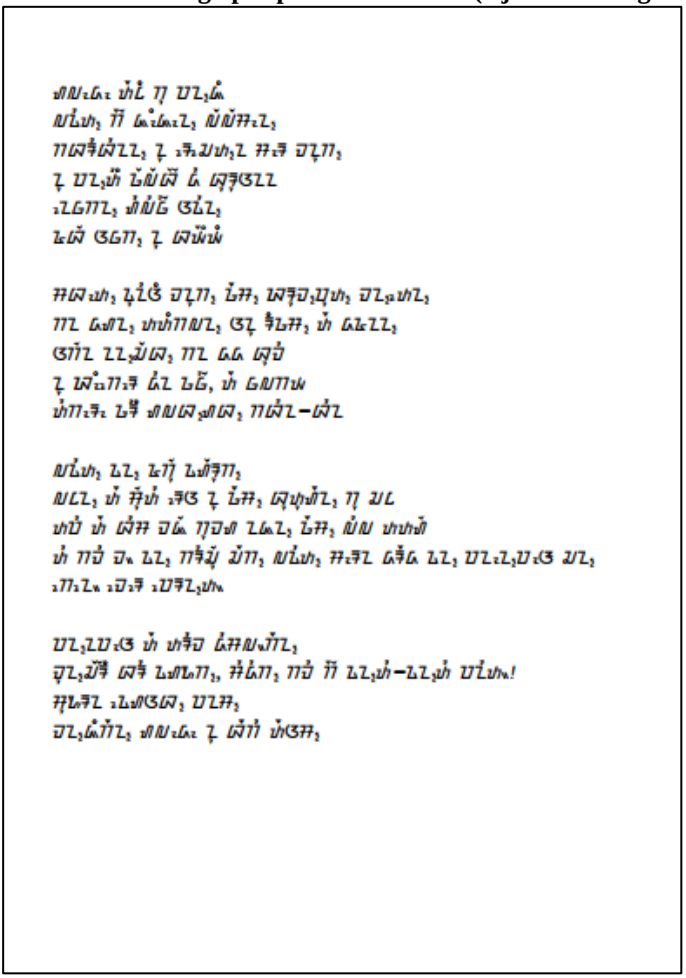
B. Masukan “Haluang Teuing ku Panjang.docx” pada Skenario A dan B (Uji Coba Fungsionalitas)



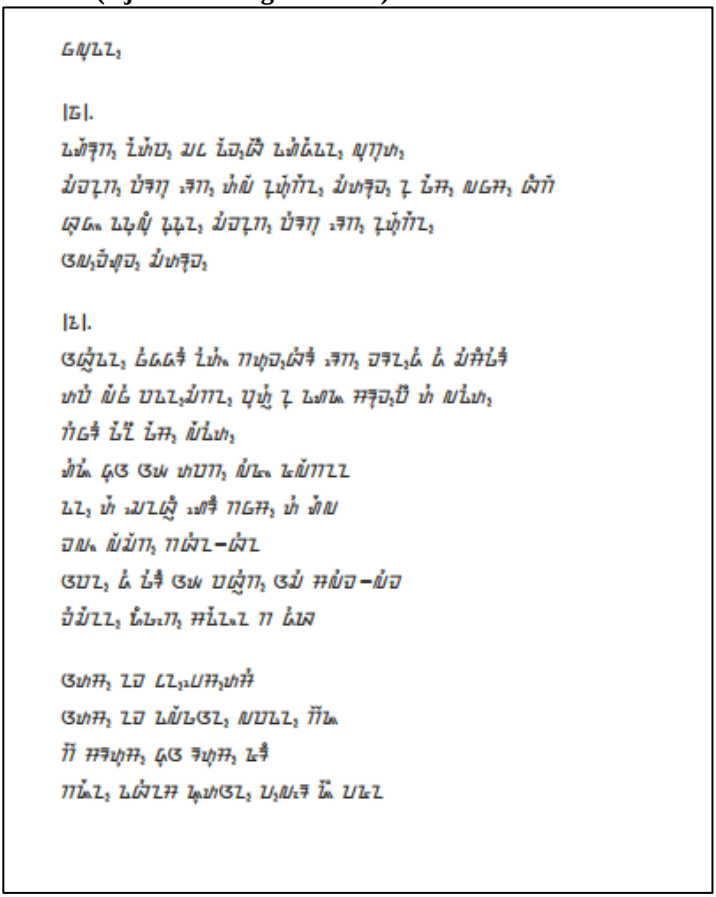
C. Hasil Stego.pdf pada Skenario A (Uji Coba Fungsionalitas)



D. Hasil Stego.pdf pada Skenario B (Uji Coba Fungsionalitas)



E. Masukan “Walungan.docx” pada Skenario C dan D (Uji Coba Fungsionalitas)



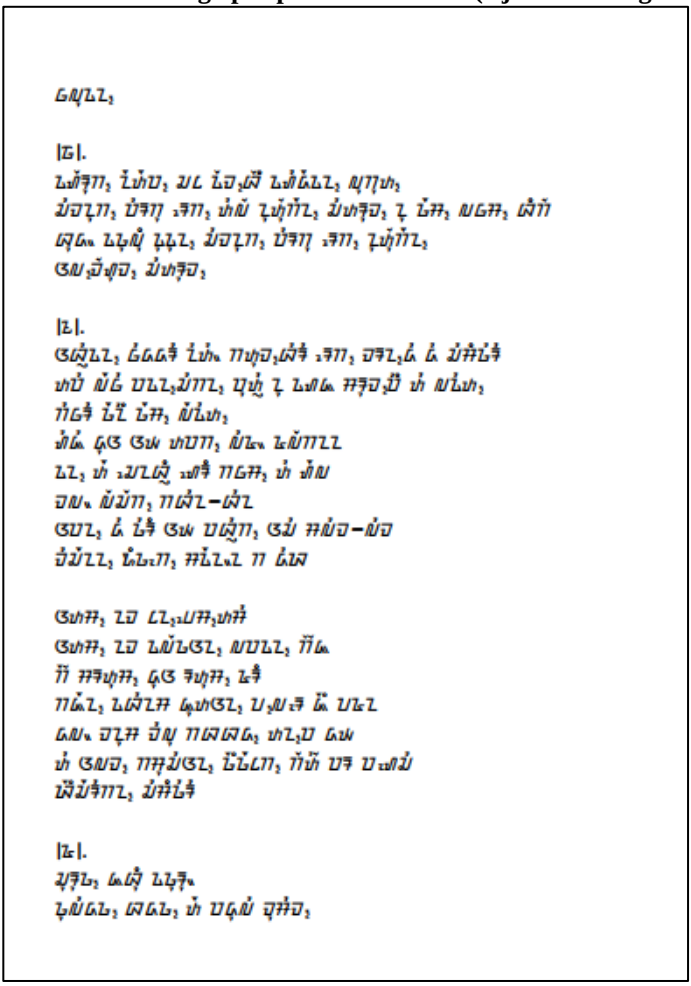
ꬻꬹꬷ ꬲꬶꬵ ꬳꬺꬹ ꬵꬻꬻꬻꬷ ꬴꬶꬷꬹ ꬻꬹ
ꬴꬷ ꬽꬹꬲꬷ ꬵꬵꬹꬽꬽꬶ ꬶꬶꬴꬽ ꬲꬶ ꬹꬷ
ꬺꬽꬳꬵꬵꬶ ꬽꬴꬳꬶ

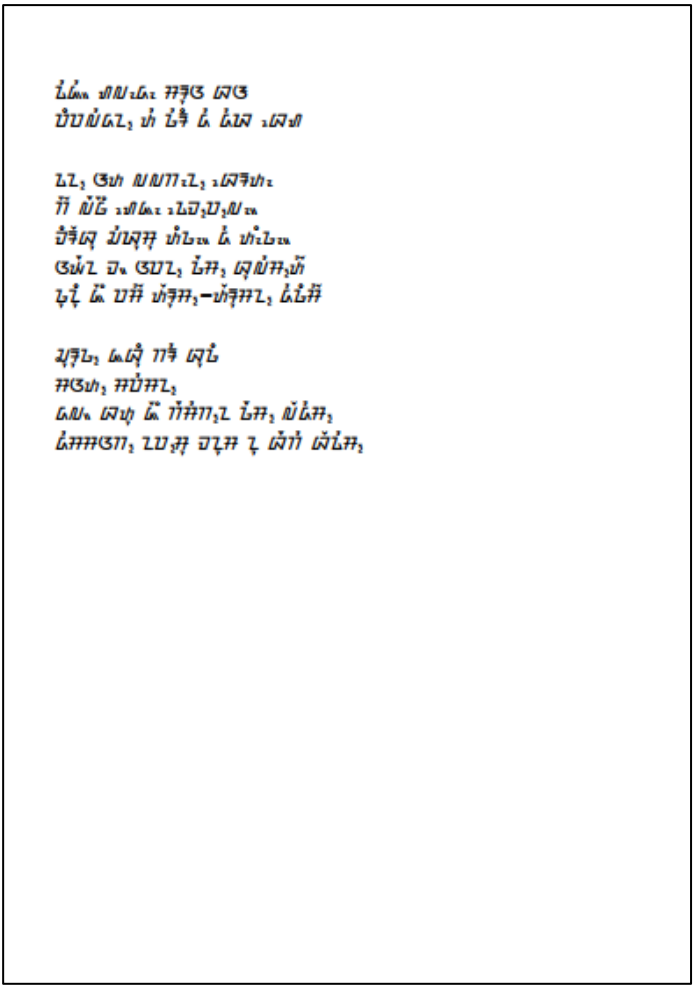
|ꬸ|.
ꬾꬵꬶ ꬻꬾꬳ ꬶꬸꬵ
ꬸꬽꬻꬶ ꬻꬻꬶ ꬴ ꬹꬸꬽ ꬳꬵꬲ
ꬶꬽꬷ ꬴꬹꬻꬸ ꬵꬵꬻ
ꬹꬽꬷꬷ ꬴꬹꬻꬸ

ꬶꬶ ꬽꬴ ꬹꬹꬵꬻꬶ ꬻꬻꬵꬴꬻ
ꬵꬵ ꬹꬽꬺ ꬻꬴꬷꬻ ꬻꬶꬽꬹꬷꬻ
ꬳꬵꬻ ꬽꬴꬸꬵ ꬴꬶꬽꬻ ꬸ ꬴꬻꬽꬻ
ꬽꬾꬶ ꬳ ꬽꬹꬶ ꬾꬵꬵ ꬻꬳꬵꬾꬾ
ꬸꬷ ꬳꬸ ꬽꬹꬶ ꬶꬵ ꬻꬳꬵꬾꬾ

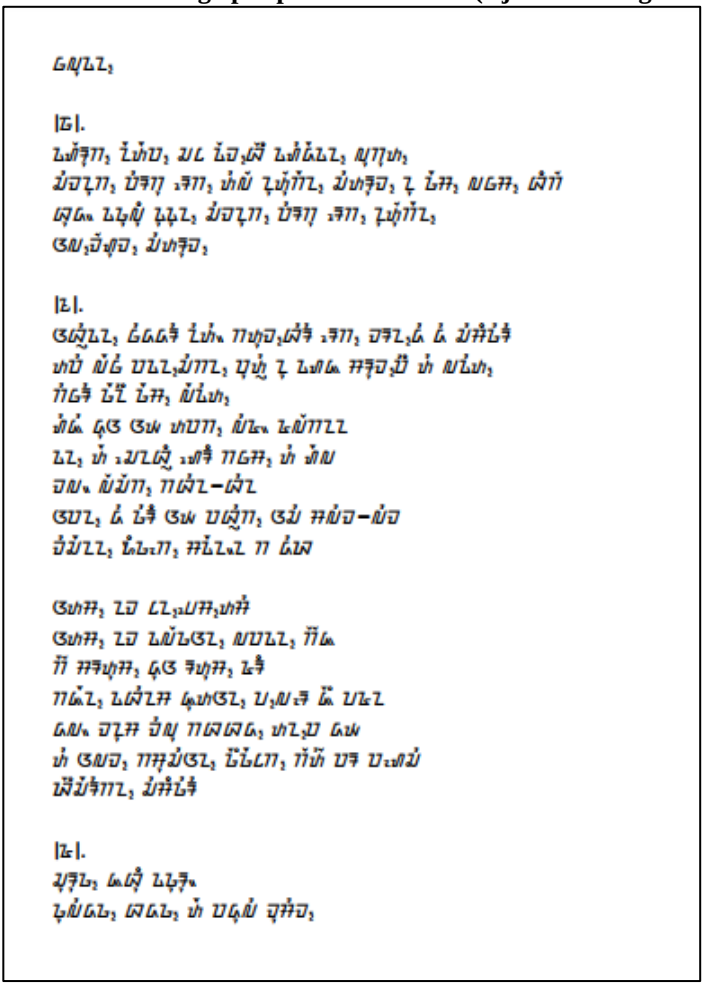
ꬾꬵꬶ ꬻꬾꬳ ꬵꬵ ꬻꬳꬶ
ꬵꬽꬴ ꬵꬽꬴꬶ
ꬻꬹꬷ ꬻꬳ ꬺꬵꬵꬶ ꬶꬵ ꬹꬽꬻꬵ
ꬻꬵꬵꬽꬵꬵ ꬶꬹꬽꬵ ꬲ ꬻꬵꬴ ꬻꬶꬻꬵ

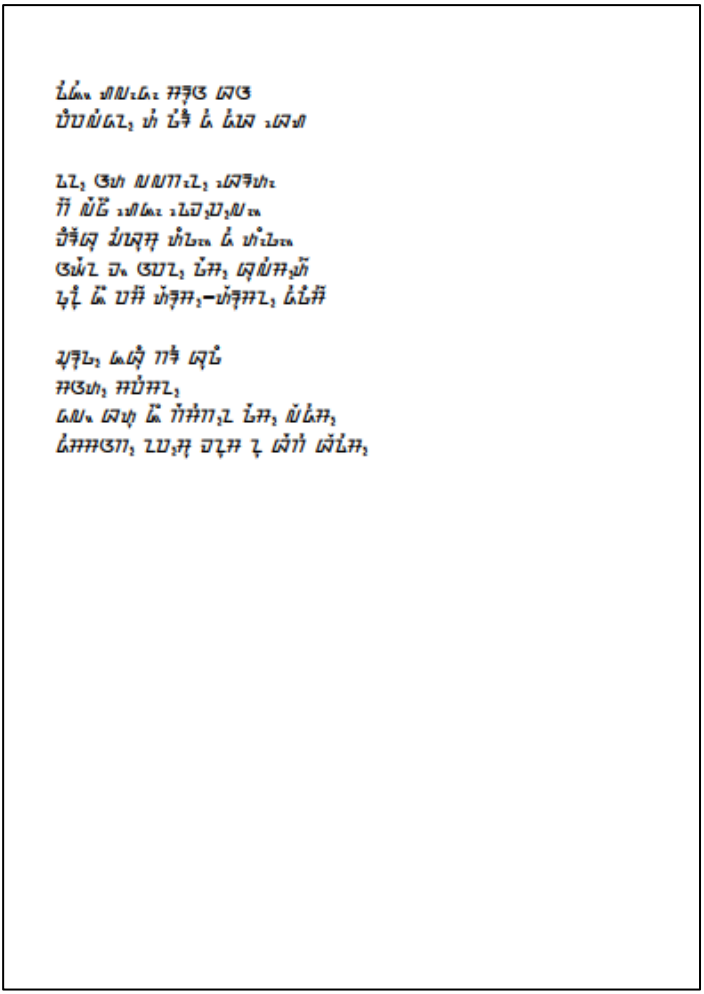
F. Hasil Stego.pdf pada Skenario C (Uji Coba Fungsionalitas)





G. Hasil Stego.pdf pada Skenario D (Uji Coba Fungsionalitas)





H. Perbandingan Ukuran Teks Percobaan 1 (A, B, C, D)

Data	Cover (kB)	Stego (kB)			
		A	B	C	D
1	7,17	7,56	7,53	7,60	7,54
2	8,24	8,66	8,65	8,72	8,69
3	6,92	7,32	7,31	7,33	7,39
4	8,48	9,02	8,95	9,02	9,02
5	8,59	9,05	9,08	9,06	9,13
6	14,78	15,71	15,70	15,68	15,74
7	19,20	20,75	20,79	20,78	20,86
8	8,96	9,52	9,52	9,53	9,49
9	9,88	10,45	10,37	10,38	10,42
10	16,62	17,69	17,73	17,73	17,70

I. Perbandingan Ukuran Teks Percobaan 2 (A, B, C, D)

Data	Cover (kB)	Stego (kB)			
		A	B	C	D
1	7,17	7,50	7,64	7,56	7,62
2	8,24	8,70	8,66	8,67	8,68
3	6,92	7,34	7,29	7,33	7,38
4	8,48	8,98	8,98	9,01	9,00
5	8,59	9,03	9,05	9,05	9,03
6	14,78	15,67	15,72	15,73	15,72
7	19,20	20,76	20,85	20,76	20,85
8	8,96	9,51	9,52	9,48	9,53
9	9,88	10,38	10,37	10,43	10,39
10	16,62	17,7	17,71	17,7	17,72

J. Perbandingan Ukuran Teks Percobaan 3 (A, B, C, D)

Data	Cover (kB)	Stego (kB)			
		A	B	C	D
1	7,17	7,64	7,60	7,64	7,60
2	8,24	8,64	8,71	8,70	8,67
3	6,92	7,31	7,30	7,33	7,36
4	8,48	9,02	9,02	9,04	9,03
5	8,59	8,99	9,02	9,12	9,12
6	14,78	15,69	15,76	15,72	15,72
7	19,20	20,76	20,83	20,76	20,81
8	8,96	9,50	9,48	9,47	9,50
9	9,88	10,37	10,39	10,40	10,40
10	16,62	17,67	17,70	17,71	17,73

K. Perbandingan Ukuran Teks Semua Skenario

Data	Cover (kB)	Stego (kB)							
		A	B	C	D	E	F	G	H
1	7,17	7,57	7,59	7,60	7,59	7,46	7,58	7,57	7,57
2	8,24	8,67	8,67	8,70	8,68	8,60	8,53	8,66	8,60
3	6,92	7,32	7,30	7.,33	7,38	7,27	7,28	7,33	7,36
4	8,48	9,01	8,98	9,02	9,02	8,76	8,78	8,87	8,89
5	8,59	9,02	9,05	9,08	9,10	8,83	8,88	8,97	8,98
6	14,78	15,69	15,73	15,71	15,73	15,14	15,14	15,18	15,18
7	19,20	20,76	20,82	20,77	20,84	19,54	19,51	19,66	19,71
8	8,96	9,51	9,51	9,49	9,51	9,23	9,23	9,34	9,37
9	9,88	10,4	10,38	10,4	10,4	10,10	10,18	10,28	10,26
10	16,62	17,69	17,71	17,71	17,72	17,00	16,97	17,11	17,07

L. Perubahan Ukuran Teks

Data	Cover (kB)	Perubahan Ukuran Teks (kB)							
		A	B	C	D	E	F	G	H
1	7,17	0,40	0,42	0,43	0,42	0,29	0,41	0,40	0,40
2	8,24	0,43	0,43	0,46	0,44	0,36	0,29	0,42	0,36
3	6,92	0,40	0,38	0,41	0,46	0,35	0,36	0,41	0,44
4	8,48	0,53	0,50	0,54	0,54	0,28	0,30	0,39	0,41
5	8,59	0,43	0,46	0,49	0,51	0,24	0,29	0,38	0,39
6	14,78	0,91	0,95	0,93	0,95	0,36	0,36	0,40	0,40

7	19,20	1,56	1,62	1,57	1,64	0,34	0,31	0,46	0,51
8	8,96	0,55	0,55	0,53	0,55	0,27	0,27	0,38	0,41
9	9,88	0,52	0,50	0,52	0,52	0,22	0,30	0,40	0,38
10	16,62	1,07	1,09	1,09	1,10	0,38	0,35	0,49	0,45
Rata-rata		0,68	0,69	0,70	0,71	0,31	0,32	0,41	0,42
Nilai Maksimum		1,56	1,62	1,57	1,64	0,38	0,41	0,49	0,51
Nilai Minimum		0,40	0,38	0,41	0,42	0,22	0.27	0.38	0,36

M. Waktu Embedding Percobaan 1

Data	Cover (kB)	Waktu *Embedding* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	10,28	10,19	10,65	9,04	9,44	9,76	10,59	10,17
2	8,24	4,98	5,10	5,19	4,58	3,71	4,60	4,91	4,92
3	6,92	11,33	9,57	11,14	9,91	8,39	9,22	10,52	10,64
4	8,48	18,61	21,62	25,52	17,31	17,09	14,94	15,57	20,8
5	8,59	13,26	11,42	11,62	10,34	9,55	10,85	10,25	10,94
6	14,78	174,27	123,45	152,98	134,22	108,85	119,33	120,52	156,98
7	19,20	543,15	495,37	529,85	442,75	421,79	443,24	444,64	440,02
8	8,96	17,37	16,13	16,83	15,87	14,44	15,57	15,28	15,76
9	9,88	23,17	20,28	20,14	20,71	15,67	14,85	15,12	16,74
10	16,62	150,12	138,27	155,13	143,09	143,46	151,46	153,33	152,18

N. Waktu Embedding Percobaan 2

Data	Cover (kB)	Waktu *Embedding* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	10,21	9,48	11,65	9,86	8,44	9,23	10,51	9,86
2	8,24	4,81	4,99	5,25	4,73	4,05	4,50	4,49	4,57
3	6,92	8,90	9,13	10,69	10,18	8,46	8,62	9,87	10,44
4	8,48	19.63	14,92	17,81	24,89	17,2	15,75	18,87	19,81
5	8,59	13,10	11,88	12,45	10,1	9,71	11,86	11,29	11,92
6	14,78	173,15	143,75	179,49	135,25	106,66	121,96	118,83	128,80
7	19,20	547,73	477,14	525,85	441,58	432,68	422,54	440,95	471,85
8	8,96	17,94	16,44	16,92	16,13	15,09	15,87	15,67	16,22
9	9,88	23,18	21,6	27,52	21,27	14,97	14,74	17,81	18,92
10	16,62	155,98	135,95	151,27	156,65	125,11	143,86	144,44	142,09

O. Waktu Embedding Percobaan 3

Data	Cover (kB)	Waktu *Embedding* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	9,59	9,71	10,23	10,05	8,84	9,45	10,47	10,08
2	8,24	5,35	4,51	4,89	4,77	4,08	4,55	4,99	5,31
3	6,92	8,83	10,44	10,33	9,84	8,65	8,47	10,45	13,03
4	8,48	15,42	17,24	20,16	19,08	13,99	14,94	14,75	14,57
5	8,59	11,87	10,78	12,34	10,47	10,22	11,41	11,98	13,23
6	14,78	140,15	123,93	143,67	122,97	116,71	122,33	120,42	123,8
7	19,20	555,93	472,66	535,34	431,93	422,77	425,54	446,64	487,27

8	8,96	19,03	17,97	16,68	15,05	14,96	15,00	15,37	15,37
9	9,88	20,83	21,52	23,93	25,76	15,14	14,68	14,9	16,77
10	16,62	158,76	149,38	160,98	162,62	127,00	136,33	156,44	160,72

P. Waktu Embedding Rata-rata (Percobaan 1, 2, dan 3)

Data	Cover (kB)	Waktu *Embedding* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	10,03	9,79	10,84	9,65	8,91	9,48	10,52	10,04
2	8,24	5,05	4,87	5,11	4,69	3,95	4,55	4,80	4,93
3	6,92	9,69	9,71	10,72	9,98	8,50	8,77	10,28	11,37
4	8,48	17,89	17,93	21,16	20,43	16,09	15,21	16,4	18,39
5	8,59	12,74	11,36	12,14	10,3	9,83	11,37	11,17	12,03
6	14,78	162,52	130,38	158,71	130,81	110,74	121,21	119,92	136,53
7	19,20	548,94	481,72	530,35	438,75	425,75	430,44	444,08	466,38
8	8,96	18,11	16,85	16,81	15,68	14,83	15,48	15,44	15,78
9	9,88	22,39	21,13	23,86	22,58	15,26	14,76	15,94	17,48
10	16,62	154,95	141,2	155,79	154,12	131,86	143,88	151,4	151,66
Rata-rata		96,23	84,49	94,55	81,7	74,57	77,52	80,00	84,46
Nilai Maks.		548,94	481,72	530,35	438,75	425,75	430,44	444,08	466,38
Nilai Min.		5,05	4,87	5,11	4,69	3,95	4,55	4,80	4,93

Q. Waktu Extracting Percobaan 1

Data	Cover (kB)	Waktu *Extracting* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	402,99	403,37	1205,72	1204,58	403,19	402,95	1205,29	1204,53
2	8,24	402,80	402,76	1204,52	1204,23	403,69	402,82	1205,57	1204,52
3	6,92	403,41	403,10	1204,92	1204,38	402,99	402,77	1204,93	1204,55
4	8,48	403,81	402,75	1204,60	1203,82	403,01	402,57	1204,87	1203,95
5	8,59	402,73	401,14	1205,15	1204,18	402,86	402,81	1204,70	1204,32
6	14,78	402,87	402,97	1204,19	1204,22	403,11	403,85	1204,45	1204,79
7	19,20	402,99	402,84	1204,29	1204,78	403,02	402,79	1203,51	1204,40
8	8,96	402,85	402,56	1204,59	1203,41	402,72	402,85	1204,54	1204,45
9	9,88	402,80	402,61	1204,85	1204,25	402,70	402,72	1204,69	1204,29
10	16,62	402,67	402,66	1204,79	1204,03	402,76	402,68	1204,67	1203,97

R. Waktu Extracting Percobaan 2

Data	Cover (kB)	Waktu *Extracting* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	402,88	402,57	1204,55	1204,55	403,15	403,46	1204,67	1204,67
2	8,24	403,01	402,86	1204,83	1204,55	402,83	403,16	1204,94	1204,61
3	6,92	402,84	402,40	1204,43	1204,39	402,96	402,96	1203,43	1204,40
4	8,48	402,44	402,56	1204,35	1204,41	402,56	402,61	1204,80	1203,78
5	8,59	402,57	402,94	1206,14	1204,54	402,39	402,79	1204,98	1204,53
6	14,78	402,79	403,08	1204,67	1204,04	402,76	402,77	1204,76	1203,18
7	19,20	402,65	402,89	1204,70	1204,92	403,09	402,77	1204,33	1204,64

8	8,96	402,66	402,53	1204,65	1204,34	402,98	402,85	1204,60	1205,16
9	9,88	402,81	402,64	1205,09	1204,17	402,77	402,82	1204,75	1205,31
10	16,62	403,02	402,71	1205,27	1204,30	402,89	403,77	1204,89	1204,15

S. Waktu Extracting Percobaan 3

Data	Cover (kB)	Waktu *Extracting* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	402,96	402,20	1203,29	1204.43	403,54	403,98	1205,12	1204,47
2	8,24	401,46	402,81	1202,81	1204,10	402,78	402,72	1204,66	1204,93
3	6,92	403,17	402,56	1207,00	1204,09	403,51	402,86	1205,94	1204,78
4	8,48	402,84	403,03	1204,93	1203,99	402,74	402,50	1204,30	1203,82
5	8,59	402,83	402,53	1204,42	1204,20	402,88	402,59	1205,49	1203,96
6	14,78	403,35	402,79	1204,45	1203,98	403,66	402,86	1204,56	1204,59
7	19,20	402,96	402,55	1204,95	1204,64	402,84	403,12	1203,47	1203,98
8	8,96	402,82	403,28	1204,35	1204,49	402,75	402,91	1204,60	1204,23
9	9,88	403,30	402,53	1204,58	1205,32	403,92	402,57	1204,44	1204,03
10	16,62	402,94	402,95	1204,60	1204,62	402,69	402,16	1204,47	1205,14

T. Waktu Extracting Rata-rata

Data	Cover (kB)	Waktu *Extracting* (*miliseconds*)							
		A	B	C	D	E	F	G	H
1	7,17	402,94	402,71	1204,52	1204,52	403,29	403,46	1205,03	1204,56
2	8,24	402,42	402,81	1204,05	1204,29	403,10	402,90	1205,06	1204,69
3	6,92	403,14	402,69	1205,45	1204,29	403,15	402,86	1204,77	1204,58
4	8,48	403,03	402,78	1204,63	1204,07	402,77	402,56	1204,66	1203,85
5	8,59	402,71	402,20	1205,24	1204,31	402,71	402,73	1205,06	1204,27
6	14,78	403,00	402,95	1204,44	1204,08	403,18	403,16	1204,59	1204,19
7	19,20	402,87	402,76	1204,65	1204,78	402,98	402,89	1203,77	1204,34
8	8,96	402,78	402,79	1204,53	1204,08	402,82	402,87	1204,58	1204,61
9	9,88	402,97	402,59	1204,84	1204,58	403,13	402,70	1204,63	1204,54
10	16,62	402,88	402,77	1204,89	1204,32	402,78	402,87	1204,68	1204,42
Rata-rata		402,87	402,71	1204,72	1204,33	402,99	402,90	1204,68	1204,41
Nilai Maks.		403.14	402.95	1205.45	1204.78	403.29	403.46	1205.06	1204.69
Nilai Min.		402.42	402.2	1204.05	1204.07	402.71	402.56	1203.77	1203.85

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Mohammad Rijal lahir di Tasikmalaya, 12 Desember 1995. Penulis merupakan anak dari pasangan H. Dudung Abdul Halim dan Hj. Nunuy Nurhayati. Pendidikan formal yang ditempuh mulai dari MI Cipasung II (2001-2007), SMP Islam Cipasung (2007-2010), MAN Model Cipasung (2010-2013) dan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember. Selain itu, penulis pernah mendapatkan beasiswa untuk belajar di Cardiff Metropolitan University, UK (2016-2017) selama satu semester. Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ).

Selama berkuliah di Teknik Informatika ITS, penulis aktif dalam kegiatan kemahasiswaan dan kepemanduan. Penulis merupakan Pemandu Titan FTIf ITS dan pernah menjabat sebagai Manager Information Media Department di Badan Eksekutif Mahasiswa FTIf ITS (2015-2016). Penulis juga aktif dalam kepanitiaan Schematics pada Divisi National Programming Contest (2016) sebagai Staf Ahli. Penulis dapat dihubungi melalui email **mohammad.rijal@outlook.com.**